Vision-Aided, Cooperative Navigation for Multiple Unmanned Vehicles

THESIS

Jason K. Bingham, Captain, USAF

AFIT/GE/ENG/09-05

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GE/ENG/09-05

Vision-Aided, Cooperative Navigation for Multiple Unmanned Vehicles

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Jason K. Bingham, BSEE

Captain, USAF

March 2009

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT/GE/ENG/09-05

Vision-Aided, Cooperative Navigation for Multiple Unmanned Vehicles

Jason K. Bingham, BSEE
Captain, USAF

Approved:

/signed/	16 March 2009
LtCol Michael Veth, PhD (Chairman)	Date
/signed/	16 March 2009
Dr. John F. Raquet (Member)	Date
/signed/	16 March 2009
Dr. Mark E. Oxley (Member)	Date

AFIT/GE/ENG/09-05

Abstract

In the recent past, a shift has taken place from manned to unmanned Intelligence, Surveillance, and Reconnaissance (ISR) missions. This shift has lead to an increase in the number of unmanned vehicles (UV) operating in a theater. Additionally, removal of the crew allows for a reduction in vehicle scale, which leads to an increased ability to operate in GPS degraded environments. With the loss of GPS signals the vehicles must rely on Inertial Navigation Systems (INS) which when reduced to an appropriate size are inherently inaccurate. This research endeavors to exploit three attributes of increased UV use for ISR missions. These attributes are: increased numbers of UVs, on-board vision, and wireless communications.

This research's focus is the development and validation of a cooperative navigation system based on the measurement of UV position relative to shared landmark position estimates. Each UV in the network locates landmarks using it's on-board vision system and transmits the data to all other system UVs. After receiving data from the other UVs, the system fuses the landmarks with on-board measurements using a federated filter architecture.

The system is evaluated using `Matlab`® simulation. Simulations of the cooperative system, with and without ranging, are compared to a non-cooperative simulation. The comparison is performed using four platform motion scenarios: stationary, linear, angular, and full motion. The simulation results demonstrate position error estimate improvements of $0.5\,cm$ to $1\,cm$. Additionally, the stationary and linear motion scenarios demonstrate attitude observability difficulties eliminated by the introduction of angular motion.

Table of Contents

Page

List of Figures

Figure Page

Figure Page

List of Tables

List of Abbreviations

Abbreviation		Page
UAV	Unmanned Aerial Vehicle	1
ISR	Intelligence, Surveillance, and Reconnaissance	1
UGV	Unmanned Ground Vehicle	1
UMV	Unmanned Maritime Vehicle	1
UV	Unmanned Vehicle	1
GPS	Global Positioning System	2
INS	Inertial Navigation System	2
DoD	Department of Defense	2
CNS	Cooperative Navigation System	4
MEMS	Micro-miniature Electromechanical Systems	6
DCM	Direction Cosine Matrix	7
EKF	Extended Kalman Filter	20
SIFT	Scale-Invariant Feature Tracking	22
RSS	Root-Sum-Squared	49
RMS	Root-Mean-Squared	57

Vision-Aided, Cooperative Navigation for Multiple Unmanned Vehicles

I. Introduction

This thesis outlines a research effort focused on expanding previous research into the fusion of optical and inertial sensors for robust, autonomous navigation to multiple platforms. This research effort is motivated by the recent increase in the number of small unmanned vehicles with the ability to operate in environments where external navigation reference sources are unavailable.

The development of Unmanned Aerial Vehicles (UAVs) began soon after the first manned flight. Originally, this progression was constrained to flying bomb or cruise missile design and testing. However, UAV development quickly expanded into Intelligence, Surveillance, and Reconnaissance (ISR) missions following the 1959 shoot-down of Francis Gary Powers over Russia [4]. In addition to advancements in UAV technology, numerous Unmanned Ground (UGVs) and Maritime (UMVs) vehicles have been developed for ISR work. The evolution of unmanned systems has lead to a shift from manned to unmanned ISR missions which has in-turn lead to an increase in the number of Unmanned Vehicles (UVs) operating in a theater.

“Today, we now have more than 5,000 UAVs, a 25-fold increase since 2001. But in my view, we can do - and we should do - more to meet the needs of men and women fighting in the current conflicts...” [6] With this statement, during a speech to the Air War College in April 2008, Defense Secretary Gates called for further increase in the number of UAVs fielded by the Air Force.

The development of UV technologies has removed the pilot and crew from the vehicle to safe locations. As a result the scale of these vehicles may be reduced considerably. This reduction in scale allows for the increased operation of UVs in Global

Positioning System (GPS) degraded environments. These environments include but are not limited to, urban canyons, caves, underwater, and building interiors. With the loss of GPS signal the vehicles are forced to rely on Inertial Navigation Systems (INS) which, when reduced to an appropriate size, are inherently inaccurate.

As the reliance on these vehicles increases so does the desire for ever more accurate navigation and targeting. "The ability to positively identify and precisely locate military targets in real-time is a current shortfall with DoD UAS." In response, the Department of Defense (DoD) has designated reconnaissance and precision targeting as two of their top three priorities for UVs in all three categories (UAV, UGV, UMV) [14]. The ability to precisely locate targets is dependent on the accuracy of the targeting platform's navigation state. Therefore, any increase in the accuracy of the navigation state also improves the location estimate of a target.

The concept of this thesis is inspired by three factors. First, previous research to exploit the availability of visual data on board a UV demonstrates the ability of vision aiding to improve the navigation solution. Next, the increased numbers of UVs operating within a theater provides for multiple sources of visual data. Finally, the capability of wireless communication, inherent in the vast majority of UVs designed today, provides a readily available means of sharing visual data between platforms.

1.1 Problem Definition

The measurement of a target location is a function of the position of the measurement device, the alignment of the measurement device to the platform, and the accuracy of the measurement device. Therefore, the accuracy of the navigation solution impacts the ability of the platform to precisely locate targets. This research focuses on the accuracy of the navigation solution and leaves the alignment and accuracy of the measurement device to other research efforts.

For the purposes of this thesis, navigation is defined as the determination of position, velocity, and attitude relative to a predefined reference frame. The most common method for determining a navigation solution, in the absence of GPS, is the INS.

1.1.1 Inertial Navigation Systems.

An INS is composed of two types of passive sensors: accelerometers and gyroscopes. Accelerometers measure the specific force acting on the platform in the sensitive axis direction of the accelerometer, while gyroscopes measure the angular rotation rate of the platform with respect to an inertial reference frame. The measurements of both devices are corrupted by errors which are discussed further in Chapter II, Sections 2.4.1 and 2.4.2 respectively. The combination of measurements from these two devices provide a navigation solution relative to an initial position.

The solution of an INS will drift away from the true solution as a function of time, due to the errors inherent in the sensors. This drift may be reduced by the incorporation of additional measurements to the system. Previous research has shown visual aiding of an INS reduces this drift.

1.1.2 Vision Aiding.

Vision aiding is the use of image sensors to measure the position of features within a sequence of images to measure the linear and angular drift of the platform. Previous research into vision aiding of an INS is discussed further in Chapter II, Section 2.6.

1.2 Research Contribution

The primary contribution of this research is the application of distributed filtering techniques to existing research into the coupling of image and inertial sensors. This expands the current capabilities of vision-aided navigation to multiple platform cooperative systems, improving the accuracy of individual platform navigation solutions.

1.3 Assumptions

This research is an extrapolation of the image and inertial fusion navigation algorithm developed in [20]. Therefore, the following four assumptions of the algorithm are required for this research:

- the system includes a strapdown inertial sensor which is rigidly mounted relative to one or more cameras,
- the cameras capture images at known times relative to inertial measurements,
- the initial navigation state and accuracy statistics are known, and
- the estimated distance to objects in the scene are available.

The estimated distance to objects may be provided from exploitation of knowledge of the world or through binocular stereopsis.

In addition to these assumptions, the multiple platform nature of this research requires:

- image feature descriptors generated on each platform are of the same format,
- feature positions are computed in the n-frame,
- a common communication protocol is available to all platforms for the transfer of data, and
- to simplify implementation of the cooperative navigation system (CNS), range measurements between platforms are acquired at the same moment as a platform position is generated.

These additional assumptions are discussed in detail in Chapter III.

1.4 Overview of Thesis

This thesis is organized as follows. Chapter II provides the mathematical background for the cooperative, multiple platform vision aided navigation system. This includes mathematical notation, coordinate frames and transformations, inertial navigation, extended Kalman and federated filtering, and previous vision aiding and multiple vehicle research. Chapter III describes the methodology used to develop the cooperative vision aided navigation system. Chapter IV presents the simulation data and analyzes the results. Finally, Chapter V presents conclusions regarding the theory and recommendations for future research.

II. Background

This chapter reviews the concepts and background required to fully develop the problem of multiple unmanned vehicle, cooperative, vision-aided navigation. The chapter opens with a definition of the mathematical notation used throughout the document. Next, the reference frames used for inertial navigation are defined. A short discussion of coordinate transformations presents the mathematical techniques required to properly format coordinates for any reference frame. Then, the concepts of INS operation are briefly discussed with an emphasis on Micro-miniature Electromechanical Systems (MEMS) units. A brief discussion of Kalman filtering follows, providing a foundation for vision-aiding, federated filtering, and this research. Following this, the background of INS vision-aiding and the work performed in this field are explored to develop the vision framework of the problem. Next, the federated filter is defined to provide the architecture upon which the sharing of information, and navigation state estimates are developed. Finally, the concept of multiple vehicle cooperation is presented to provide a foundation for the development of a network of unmanned agents.

2.1 Mathematical Notation

The mathematical notation used throughout this research is described in Table 2.1.

2.2 Reference Frames

Precise definition of coordinate reference frames is a vital prerequisite for inertial navigation. For the purposes of this thesis, five reference frames are defined based on [7] and [19]. All of the reference frames defined are right-handed orthonormal coordinated systems in $\Re^3$.

The true inertial reference frame (*I-frame*) has no fixed origin or orientation; is non-rotating and non-accelerating. This is a theoretical reference frame in which the Newtonian laws of physics apply.

Table 2.1: Mathematical Notation.

Type	**Description**	**Example**
Scalars	Scalar variables are designated with italics.	x or X
Vectors	Vectors are denoted by lower case bold.	$\mathbf{x}$
Matrices	Matrices are denoted by uppercase bold.	$\mathbf{X}$
Transpose	The transpose of a matrix or vector is designated with a superscript capital T.	$\mathbf{x}^T$ or $\mathbf{X}^T$
Estimates	Estimates of random variables are identified with the *hat* character.	$\hat{\mathbf{x}}$
Calculated Variables	Variables corrupted by errors are denoted with the *tilde* character.	$\tilde{\mathbf{x}}$
Nominal Values	Nominal values are denoted with a bar.	$\bar{\mathbf{x}}$
Direction Cosine Matrix (DCM)	DCMs are designated by a bold capital C with a subscript designating the originating coordinate frame and a superscript designating the resulting coordinate frame.	$\mathbf{C}_o^r$
Frame of Reference	Vectors expressed in a specific reference frame are annotated with a superscript letter representing the frame.	$\mathbf{p}^n$
Skew Symmetric Form	Vectors represented in skew symmetric form are enclosed in parentheses with a cross product symbol.	$(\mathbf{x}\times)$

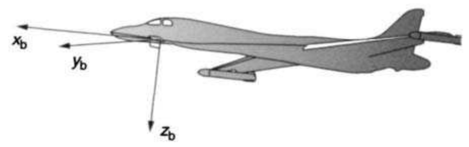


Figure 2.1: Body Frame of Reference [19]

The inertial frame (*i-frame*) is a non-rotating, Earth-centered frame in which the z-axis is coincident with the polar axis. Due to the motion of the Earth around the Sun, and the motion of the Sun through space, the *i-frame* is an accelerating frame. Additionally, the *i-frame* is non-rotational with respect to the fixed stars. As the *I-frame* has no fixed origin or orientation, the *i-frame* is designed to provide these critical components of a reference frame. This allows measured physical values to be transformed into other frames of reference as discussed in Section 2.3.

The Earth frame (*e-frame*) originates at the center of the Earth and is fixed with respect to the surface. The z-axis is aligned with the geographic polar axis. The x-axis lies along the intersection of the Greenwich meridian and equatorial planes. The *e-frame* therefore rotates in relation to the *i-frame*.

The navigation frame (*n-frame*) is centered about a fixed point on the navigating body. The x-axis lies within the plane created by the Earth's polar axis and the origin of the *n-frame*, pointing towards the north pole. The z-axis points in the direction of gravitational acceleration.

The body frame (*b-frame*) is centered at the origin of the *n-frame*. However, the axes differ from the *n-frame*. The axes are aligned with the roll, pitch and yaw axes of an aircraft as depicted in Figure 2.1.

In addition to these five reference frames, a system designer may develop any new arbitrary reference frame which will simplify the data processing for the system. For the purpose of this thesis, the set of these possible frames will be annotated as the l_i*-frame* local frames, or as c_i*-frame* for camera frames, where $i = 1 \dots n$ and n is the total number of local or camera frames.

The next section discusses the means to transform coordinates between these coordinate systems.

2.3 Coordinate Transformations

As discussed in [19] coordinate transformations describe the relationship between two reference frames and are classified as either three or four-parameter transformations. Three-parameter coordinate transformations contain a singularity at a pitch angle of 90°. Therefore, this research will use the four-parameter DCM coordinate transformation.

The DCM is a 3x3 matrix representing the unit vector of the originating frame projected along the axis of the resulting frame. The DCM is written in component form as:

$$\mathbf{C}_o^r = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{2.1}$$

The elements c_{ij} represents the cosine of the angle between the i-axis of originating reference frame and the j-axis of the resulting frame.

The DCM is propagated in time through:

$$\dot{\mathbf{C}}_o^r = \mathbf{C}_o^r \mathbf{\Omega}_{ro}^o \tag{2.2}$$

where

$$\mathbf{\Omega}_{ro}^o = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.3}$$

is the skew symmetric form of the angular rate vector $\boldsymbol{\omega}_{ro}^o = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$, which represents the angular turn rate of the originating frame with respect to the resulting frame expressed in the axes of the originating frame.

In the absence of a GPS signal, an INS may provide the coordinates for transformation.

2.4 Inertial Navigation

In this section, the basic concepts of inertial navigation are discussed. This discussion includes strapdown INS sensors, related errors, available measurements, and differential error equations.

2.4.1 Accelerometers. Acceleration due to inertial motion and acceleration due to gravity are indistinguishable as described by Einstein's Theory of Special Relativity [5]. Therefore, accelerometers measure the specific force applied to them in the direction of alignment. This measurement is the difference between the inertial motion and gravitational accelerations, defined as

$$\mathbf{f} = \ddot{\mathbf{p}} - \mathbf{g} \tag{2.4}$$

where $\mathbf{f}$ is the specific force, $\mathbf{p}$ is the inertial position and $\mathbf{g}$ is the acceleration due to gravity.

The measurement may be contaminated by errors induced by the design of the accelerometer [19]. The specific force measurement ($\tilde{f}_x^b$) may be expressed as:

$$\tilde{f}_x^b = (1 + S_x) f_x^b + M_y f_y^b + M_z f_z^b + B_v f_x^b f_y^b + \mathbf{a}^b + \boldsymbol{\eta}^b \tag{2.5}$$

where f_x^b, f_y^b, f_z^b are the true specific forces applied in the sensitive and cross axes and the remaining errors are:

- **Measurement Bias**($\mathbf{a}^b$): An additive error arising as a direct result of the characteristics of the accelerometer's component parts. This error may fluctuate slowly with time and may be compensated for using mechanical calibration techniques.

- **Scale Factor**(S_x): A multiplicative error caused by temperature and non-ideal behavior of system components. As with measurement bias, scaling factors may by compensated for using calibration.
- **Cross Axis Coupling**(M_y and M_z): An additive error caused by the inclusion of forces applied to the device in the non-sensitive axes. This error may have root cause in the design of the device or in the misalignment of the device with the vehicle.
- **Vibro-pendulous Error**(B_v): An error due to vibration in the sensitive axis. This error may be reduced by isolation of the device from vibration sources.
- **Random Bias**($\boldsymbol{\eta}^b$): An error caused by instabilities within the device.

In addition to these errors, the final measurement may be corrupted by errors in the gyroscopes and inaccuracies of the gravity model used to remove the gravitational acceleration from the specific force equation.

2.4.2 Gyroscopes. A spinning wheel or rotor, by virtue of its angular momentum vector, tends to maintain the direction of its spin axis relative to inertial space [19]. This defines a reference direction which remains fixed in the *i-frame.*

The gyroscopes used in strapdown systems measure the angular rate of a platform, relative to inertial space, about the gyroscope input axis. The measurement may be contaminated by errors induced by the design of the gyroscope [19]. The angular rate measurement ($\tilde{\boldsymbol{\omega}}^b_{ib_x}$) may be expressed as:

$$\tilde{\boldsymbol{\omega}}^b_{ib_x} = (1 + S_x)\omega^b_{ib_x} + M_y\omega^b_{ib_y} + M_z\omega^b_{ib_z} + B_{gx}f^b_x + B_{gz}f^b_z + B_{axz}f^b_x f^b_z + \mathbf{b}^b + \eta^b_x \quad (2.6)$$

where $\omega^b_{ib_x}$, $\omega^b_{ib_y}$, and $\omega^b_{ib_z}$ are the true turn rates about the input, output, and spin axes respectively; f^b_x, and f^b_z are the true specific forces applied along the input and spin axes and the remaining errors are:

- **Fixed Bias** ($\mathbf{b}^b$): A sensor error present in the absence of an applied input rotation. A consequence of multiple effects, including residual torques from flexible leads, spurious magnetic fields, and temperature gradients.
- **Acceleration Dependent Bias** (B_{fy} and B_{fz}): An error proportional to the magnitude of the applied acceleration. Apparent in spinning mass gyroscopes as a result of an unbalanced rotor mass.
- **Anisoelastic Bias** (B_{axz}): An error proportional to the product of acceleration along orthogonal pairs of axes. Apparent in spinning mass gyroscopes and caused by the rotor suspension structure.
- **Scale Factors** (S_x): An error caused by imperfections and temperature fluctuations in the pickoff and nulling components of the gyroscope.
- **Cross Coupling** (M_y and M_z): Errors due to the non-orthogonality of the sensor axes.
- **Zero-Mean Bias** (η_x^b): An error caused by instabilities in the gyroscope with short correlation times.

The accelerometer and gyroscope measurements equations are now used to compute the differential error equations.

2.4.3 Differential Error Equations.

The differential error equations are computed in three parts; inertial sensor errors, attitude error, and position and velocity. The results are then combined to generate the state-space error model.

2.4.3.1 Inertial Sensor Errors.

The inertial sensor errors are modeled as a bias with added random noise. The accelerometer and gyroscope measurement models are:

$$\mathbf{f}_m^b = \mathbf{f}^b + \mathbf{a}^b + \mathbf{w}_a^b \tag{2.7}$$

$$\boldsymbol{\omega}_{ib_m}^b = \boldsymbol{\omega}_{ib}^b + \mathbf{b}^b + \mathbf{w}_b^b \tag{2.8}$$

where $\mathbf{w}_a^b$ and $\mathbf{w}_b^b$ are additive white Gaussian noise processes, $\mathbf{a}^b$ is the accelerometer bias, and $\mathbf{b}^b$ is the gyroscope bias.

The accelerometer and gyroscope biases are modeled as first-order Gauss-Markov processes expressed by:

$$\dot{\mathbf{a}}^b = -\frac{1}{\tau_a}\mathbf{a}^b + \mathbf{w}^b_{a_{bias}} \tag{2.9}$$

$$\dot{\mathbf{b}}^b = -\frac{1}{\tau_b}\mathbf{b}^b + \mathbf{w}^b_{b_{bias}} \tag{2.10}$$

where τ_a and τ_b are the accelerometer and gyroscope time constants, and the processes are driven by the white noise terms $\mathbf{w}^b_{a_{bias}}$ and $\mathbf{w}^b_{b_{bias}}$.

2.4.3.2 Attitude Error. The development of the attitude differential error equations begins with the attitude error vector, which is modeled as:

$$\boldsymbol{\psi} = \begin{bmatrix} \psi_n \\ \psi_e \\ \psi_d \end{bmatrix} \tag{2.11}$$

where ψ_n, ψ_e, and ψ_d are small angles relative to the north, east and down axes of the *n-frame* respectively.

Due to the small angle assumption, the *b-frame* to *n-frame* DCM can be formulated as:

$$\tilde{\mathbf{C}}_b^n \approx [\mathbf{I} - (\boldsymbol{\psi}\times)]\mathbf{C}_b^n \tag{2.12}$$

Taking the derivative of (2.12) with respect to time and solving for $(\dot{\boldsymbol{\psi}}\times)$ results in:

$$(\dot{\boldsymbol{\psi}}\times) = [\mathbf{I} - (\boldsymbol{\psi}\times)]\mathbf{C}_b^n\boldsymbol{\Omega}_{nb}^b\mathbf{C}_n^b - \tilde{\mathbf{C}}_b^n\tilde{\boldsymbol{\Omega}}_{nb}^b\mathbf{C}_n^b \tag{2.13}$$

Next, the *b-frame* to *n-frame* rotation rate vector is defined as:

$$\tilde{\boldsymbol{\omega}}^b_{nb} = \boldsymbol{\omega}^b_{ib_m} - \tilde{\mathbf{C}}^b_n \mathbf{C}^n_e \boldsymbol{\omega}^e_{ie} \tag{2.14}$$

where $\boldsymbol{\omega}^b_{ib_m}$ is the rate vector measured by the gyroscopes.

Substituting (2.8), (2.12), and (2.14) into (2.13) and eliminating second-order terms results in the angular differential error equation:

$$\dot{\boldsymbol{\psi}} = -[(\mathbf{C}^n_e \boldsymbol{\omega}^e_{ie})\times]\boldsymbol{\psi} - \mathbf{C}^n_b \mathbf{b}^b - \mathbf{C}^n_b \mathbf{w}^b_b \tag{2.15}$$

2.4.3.3 Position and Velocity Error. The position and velocity differential error equation development begins with computing the position error.

The *n-frame* velocity is defined as:

$$\dot{\mathbf{p}}^n = \mathbf{v}^n \tag{2.16}$$

and the position differential error equation is:

$$\delta\dot{\mathbf{p}}^n = \delta\mathbf{v}^n \tag{2.17}$$

Next, the calculation of the velocity error begins with the definition of the position in the *i-frame*:

$$\mathbf{p}^i = \mathbf{C}^i_e \left[\mathbf{p}^e_0 + \mathbf{C}^e_n \mathbf{p}^n\right] \tag{2.18}$$

where $\mathbf{p}^e_0$ is the location of the origin of the local navigation frame, relative to the *e-frame*.

The acceleration, in *i-frame* coordinates, is calculated by taking the second derivative using the chain rule:

$$\ddot{\mathbf{p}}^i = \mathbf{C}^i_e \mathbf{C}^e_n \ddot{\mathbf{p}}^n + 2\mathbf{C}^i_e \boldsymbol{\Omega}^e_{ie} \mathbf{C}^e_n \dot{\mathbf{p}}^n + \mathbf{C}^i_e (\boldsymbol{\Omega}^e_{ie})^2 \left[\mathbf{p}^e_0 + \mathbf{C}^e_n \mathbf{p}^n\right] \tag{2.19}$$

Appling (2.4) to (2.19) and solving for the *n-frame* acceleration provides:

$$\ddot{\mathbf{p}}^n = \mathbf{f}^n - 2\mathbf{C}_e^n\boldsymbol{\Omega}_{ie}^e\mathbf{C}_n^e\dot{\mathbf{p}}^n - \mathbf{C}_e^n(\boldsymbol{\Omega}_{ie}^e)^2\left[\mathbf{p}_0^e + \mathbf{C}_n^e\mathbf{p}^n\right] + \mathbf{g}^n \tag{2.20}$$

Then, substituting (2.16) into (2.20) and transforming the specific force into the *b-frame* yields:

$$\dot{\mathbf{v}}^n = \mathbf{C}_b^n\mathbf{f}^b - 2\mathbf{C}_e^n\boldsymbol{\Omega}_{ie}^e\mathbf{C}_n^e\mathbf{v}^n - \mathbf{C}_e^n(\boldsymbol{\Omega}_{ie}^e)^2\left[\mathbf{p}_0^e + \mathbf{C}_n^e\mathbf{p}^n\right] + \mathbf{g}^n \tag{2.21}$$

The centripetal acceleration and gravity terms are combined using the gradient of the gravity potential:

$$W(\mathbf{p}^e) = \frac{GM}{||\mathbf{p}^e||} + \frac{1}{2}\mathbf{p}^{e^T}\boldsymbol{\Omega}_{ie}^{e^T}\boldsymbol{\Omega}_{ie}^e\mathbf{p}^e + H.O.T. \tag{2.22}$$

where GM is the gravitational constant of Earth. Substituting (2.22) into (2.21) provides:

$$\dot{\mathbf{v}}^n = \mathbf{C}_b^n\mathbf{f}^b - 2\mathbf{C}_e^n\boldsymbol{\Omega}_{ie}^e\mathbf{C}_n^e\mathbf{v}^n + \mathbf{C}_e^n\mathbf{g}^e\left[\mathbf{p}_0^e + \mathbf{C}_n^e\mathbf{p}^n\right] \tag{2.23}$$

As the velocity calculation is corrupted by accelerometer and attitude errors; substituting the position, velocity, attitude and accelerometer measurement errors equations into (2.23) yields:

$$\begin{aligned}\dot{\hat{\mathbf{v}}}^n = {} & [\mathbf{I} - (\boldsymbol{\psi}\times)]\mathbf{C}_b^n(\mathbf{f}^b + \mathbf{a}^b + \mathbf{w}_a^b) - 2\mathbf{C}_e^n\boldsymbol{\Omega}_{ie}^e\mathbf{C}_n^e(\mathbf{v}^n + \delta\mathbf{v}^n) \\ & + \mathbf{C}_e^n\mathbf{g}^e\left[\mathbf{p}_0^e + \mathbf{C}_n^e\mathbf{p}^n + \mathbf{C}_n^e\delta\mathbf{p}^n\right]\end{aligned} \tag{2.24}$$

The *n-frame* acceleration error is defined as:

$$\delta\dot{\mathbf{v}}^n = \dot{\hat{\mathbf{v}}}^n - \dot{\mathbf{v}}^n \tag{2.25}$$

Finally, the velocity differential error equation is computed by substituting (2.21) and (2.24) into (2.20) and eliminating second-order terms:

$$\delta\dot{\mathbf{v}}^n = \mathbf{C}_e^n\mathbf{G}\mathbf{C}_n^e\delta\mathbf{p}^n - 2\mathbf{C}_e^n\mathbf{\Omega}_{ie}^e\mathbf{C}_n^e\delta\mathbf{v}^n + (\mathbf{f}^n\times)\boldsymbol{\psi} + \mathbf{C}_b^n\mathbf{a}^b + \mathbf{C}_b^n\mathbf{w}_a^b \tag{2.26}$$

where $\mathbf{G}$ is the gradient of the gravity vector, which is calculated as:

$$\mathbf{G} = \frac{GM}{||\mathbf{p}^e||^3}\left[3\check{\mathbf{p}}^e(\check{\mathbf{p}}^e)^T - \mathbf{I}\right] - (\mathbf{\Omega}_{ie}^e)^2 \tag{2.27}$$

2.4.3.4 State-space Error Model. With the inertial sensor, attitude, position, and velocity differential error equations computed, the error dynamics are formulated using the linear, stochastic, state-space model which is driven by white noise:

$$\delta\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}(t)\mathbf{w}(t) \tag{2.28}$$

The navigation error state vector is composed of the position, velocity, attitude, accelerometer bias, and gyroscope bias errors, resulting in the fifteen element vector:

$$\delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{p}^n \\ \delta\mathbf{v}^n \\ \boldsymbol{\psi} \\ \delta\mathbf{a}^b \\ \delta\mathbf{b}^b \end{bmatrix}_{15\times 1} \tag{2.29}$$

The noise vector is composed of the accelerometer measurement, gyroscope measurement, accelerometer bias, and gyroscope bias noise terms, resulting in the twelve element vector:

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_a^b \\ \mathbf{w}_b^b \\ \mathbf{w}_{a_{bias}}^b \\ \mathbf{w}_{b_{bias}}^b \end{bmatrix} \tag{2.30}$$

Finally, the overall differential error equation is:

$$\delta\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{C}_e^n\mathbf{G}\mathbf{C}_n^e & -2\mathbf{C}_e^n\boldsymbol{\Omega}_{ie}^e\mathbf{C}_n^e & (\mathbf{f}^n\times) & \mathbf{C}_b^n & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & -[(\mathbf{C}_e^n\boldsymbol{\omega}_{ie}^e)\times] & \mathbf{0}_3 & -\mathbf{C}_b^n \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\frac{1}{\tau_a}\mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\frac{1}{\tau_b}\mathbf{I}_3 \end{bmatrix}\delta\mathbf{x} + \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{C}_b^n & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & -\mathbf{C}_b^n & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}\mathbf{w} \tag{2.31}$$

These equations show accelerometer and gyroscope outputs may be integrated to estimate the change in position, velocity, attitude, and inertial sensor biases. However, these estimates drift rapidly, on the order of 90 meters in 1 minute, for a standard MEMS IMU position error [10]. The drift may be constrained by incorporating measurements from additional sensors using a Kalman filter.

2.5 Kalman Filtering

This section discusses the concepts of Kalman filtering by developing the linear filter equations and expanding the linear equations to nonlinear functions.

2.5.1 Linear Kalman Filter. The linear Kalman filter is an optimal, recursive data processing algorithm [12] designed to estimate desired quantities from data provided by a noisy environment.

The general form of a time-varying linear differential equation is:

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{G}(t)\mathbf{w}(t) \tag{2.32}$$

where $\dot{\mathbf{x}}(t)$ is the state estimate, $\mathbf{F}(t)$ is the homogeneous system dynamics matrix, $\mathbf{B}(t)$ is the input matrix, $\mathbf{u}(t)$ is the system input vector, $\mathbf{G}(t)$ is the noise transformation matrix, and $\mathbf{w}(t)$ is a white Gaussian noise process of zero-mean and strength $\mathbf{Q}(t)$.

The white noise covariance is computed as:

$$E\left\{\mathbf{w}(t)\mathbf{w}^T(t+\tau)\right\} = \mathbf{Q}(t)\delta(\tau) \tag{2.33}$$

where $\delta(\tau)$ is the Dirac delta function.

The maximum *a posteriori* state estimate is the mean value and is denoted by the *hat* character:

$$\begin{aligned} \hat{\mathbf{x}}(t) &= E\{\hat{\mathbf{x}}(t)\} && (2.34) \\ &= \mathbf{\Phi}(t,t_0)E\{\hat{\mathbf{x}}(t_0)\} + \int_{t_0}^{t} \mathbf{\Phi}(t,\tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau && (2.35) \end{aligned}$$

where $\mathbf{\Phi}(t,t_0)$ is the state transition matrix which satisfies $\dot{\mathbf{\Phi}}(t,t_0) = \mathbf{F}(t)\mathbf{\Phi}(t,t_0)$ and $\mathbf{\Phi}(t_0,t_0) = \mathbf{I}$.

The mean squared value of $\hat{\mathbf{x}}(t)$ is:

$$\begin{aligned} E\left\{\hat{\mathbf{x}}(t)\hat{\mathbf{x}}^T(t)\right\} = {} & \mathbf{\Phi}(t,t_0)E\left\{\hat{\mathbf{x}}(t_0)\hat{\mathbf{x}}^T(t_0)\right\}\mathbf{\Phi}^T(t,t_0) \\ & + \int_{t_0}^{t} \mathbf{\Phi}(t,\tau)\mathbf{G}(\tau)\mathbf{Q}(\tau)\mathbf{G}^T(\tau)\mathbf{\Phi}^T(t,\tau)d\tau \end{aligned} \tag{2.36}$$

and the covariance of $\hat{\mathbf{x}}(t)$ is derived from the mean squared value by substituting

$$\begin{aligned} E\left\{\hat{\mathbf{x}}(t)\hat{\mathbf{x}}^T(t)\right\} &= \hat{\mathbf{P}}_{xx}(t) + \mathbf{m}_x(t)\mathbf{m}_x^T(t) && (2.37) \\ E\left\{\hat{\mathbf{x}}(t_0)\hat{\mathbf{x}}^T(t_0)\right\} &= \hat{\mathbf{P}}_{xx}(t_0) + \mathbf{m}_x(t_0)\mathbf{m}_x^T(t_0) && (2.38) \\ \mathbf{m}_x(t) &= \mathbf{\Phi}(t,t_0)\mathbf{m}_x(t_0) && (2.39) \end{aligned}$$

into (2.36) and solving for $\hat{\mathbf{P}}_{xx}(t)$:

$$\begin{aligned}\hat{\mathbf{P}}_{xx}(t) &= \mathbf{\Phi}(t,t_0)\hat{\mathbf{P}}_{xx}(t_0)\mathbf{\Phi}^T(t,t_0) \\ &\quad + \int_{t_0}^{t} \mathbf{\Phi}(t,\tau)\mathbf{G}(\tau)\mathbf{Q}(\tau)\mathbf{G}^T(\tau)\mathbf{\Phi}^T(t,\tau)d\tau \end{aligned} \tag{2.40}$$

Using (2.35) and (2.40) the state estimate and covariance are propagated from time t_i to t_{i+1}:

$$\hat{\mathbf{x}}(t_{i+1}^-) = \mathbf{\Phi}(t_{i+1},t_i)\hat{\mathbf{x}}(t_i^+) + \int_{t_i}^{t_{i+1}} \mathbf{\Phi}(t_{i+1},\tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau \tag{2.41}$$

$$\begin{aligned}\hat{\mathbf{P}}(t_{i+1}^-) &= \mathbf{\Phi}(t_{i+1},t_i)\hat{\mathbf{P}}(t_i^+)\mathbf{\Phi}^T(t_{i+1},t_i) \\ &\quad + \int_{t_i}^{t_{i+1}} \mathbf{\Phi}(t_{i+1},\tau)\mathbf{G}(\tau)\mathbf{Q}(\tau)\mathbf{G}^T(\tau)\mathbf{\Phi}^T(t_{i+1},\tau)d\tau \end{aligned} \tag{2.42}$$

where the superscript $-$ and $+$ represent the instant before and after a measurement update respectively.

A set of measurements $\mathbf{z}$ is a linear combination of the m measurements of interest, corrupted by an uncertain disturbance $\mathbf{v}$:

$$\mathbf{z}(t_i) = \mathbf{H}(t_i)\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{2.43}$$

where $\mathbf{H}(t_i)$ is the observation matrix and $\mathbf{v}(t_i)$ is a zero mean white Gaussian noise vector with a covariance defined as:

$$E\left\{\mathbf{v}(t_i)\mathbf{v}^T(t_i)\right\} = \mathbf{R}(t_i)\delta_{ij} \tag{2.44}$$

where δ_{ij} is the Kroeneker delta function.

The state estimates and covariance immediately after a measurement update are defined as

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)\left[\mathbf{z}(t_i) - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)\right] \tag{2.45}$$

$$\hat{\mathbf{P}}(t_i^+) = \hat{\mathbf{P}}(t_i^-) - \mathbf{K}(t_i)\mathbf{H}(t_i)\hat{\mathbf{P}}(t_i^-) \tag{2.46}$$

where $\mathbf{K}(t_i)$ is the Kalman gain matrix, defined as

$$\mathbf{K}(t_i) = \hat{\mathbf{P}}(t_i^-)\mathbf{H}^T(t_i)\left[\mathbf{H}(t_i)\hat{\mathbf{P}}(t_i^-)\mathbf{H}^T(t_i) + \mathbf{R}(t_i)\right]^{-1} \tag{2.47}$$

2.5.2 Extended Kalman Filter. The linear Kalman filter provides sufficient estimates for many real-world systems; however, the linear filter is unable to provide an adequate answer for all situations. The extended Kalman filter (EKF) [13] is developed to estimate these nonlinear cases.

The development of the EKF begins with the definition of the error model. The state estimate error model is the sum of the nominal trajectory and the error state:

$$\mathbf{x}(t) = \bar{\mathbf{x}}(t) + \delta\mathbf{x}(t) \tag{2.48}$$

The nominal trajectory is defined as:

$$\dot{\bar{\mathbf{x}}}(t) = \mathbf{f}\left[\bar{\mathbf{x}}(t), \mathbf{u}(t), t\right] \tag{2.49}$$

Applying (2.49) and the linear differential equation:

$$\delta\dot{\mathbf{x}}(t) = \mathbf{F}(t)\delta\mathbf{x}(t) + \mathbf{G}\mathbf{w}(t) \tag{2.50}$$

to (2.48) results in the nonlinear dynamics model:

$$\dot{\bar{\mathbf{x}}}(t) + \delta\dot{\mathbf{x}}(t) = \mathbf{f}\left[\bar{\mathbf{x}}(t), \mathbf{u}(t), t\right] + \mathbf{F}(t)\delta\mathbf{x}(t) + \mathbf{G}\mathbf{w}(t) \tag{2.51}$$

Then, the state estimate is defined as the sum of the nominal trajectory and the estimated error:

$$\hat{\mathbf{x}}(t) = \bar{\mathbf{x}}(t) + \delta\hat{\mathbf{x}}(t) \tag{2.52}$$

The state estimate is propagated from time t_i^+ to t_{i+1}^- by calculating the nominal trajectory with a nonlinear differential equation solver using the state estimate at t_i^+ as the initial condition. The covariance is propagated as in the linear Kalman filter.

A measurement update is performed by linearizing the nonlinear update equation:

$$\mathbf{z}(t_i) = \mathbf{h}[\mathbf{x}(t_i), t_i] + \mathbf{v}(t_i) \tag{2.53}$$

about the nominal trajectory at time t_{i+1}:

$$\mathbf{z}(t_i) = \mathbf{h}[\bar{\mathbf{x}}(t_{i+1}^-), t_{i+1}] + \mathbf{H}(t_{i+1})\delta\mathbf{x}(t_{i+1}) + \mathbf{v}(t_{i+1}) \tag{2.54}$$

where $\mathbf{H}(t)$ is

$$\mathbf{H}(t) = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}(t),t} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_n} \end{bmatrix} \tag{2.55}$$

A measurement is defined as the sum of a nominal measurement and an error:

$$\mathbf{z}(t_{i+1}) = \bar{\mathbf{z}}(t_{i+1}) + \delta\mathbf{z}(t_{i+1}) \tag{2.56}$$

Substituting (2.54) into (2.56) and solving for the measurement error provides the error update equation:

$$\delta\mathbf{z}(t_i) = \mathbf{H}(t_{i+1})\delta\mathbf{x}(t_{i+1}) + \mathbf{v}(t_{i+1}) \tag{2.57}$$

Finally, the post-measurement error state and covariance are calculated using the linear Kalman filter update equations.

The Kalman filter is unable to provide any benefit to a system without measurements of the surrounding environment. The next section discusses the use of vision systems to provide such measurements.

2.6 Vision-Aiding

In [20], Veth developed an image and inertial fusion navigation algorithm. The algorithm fuses image and inertial data by, *a*) collecting an image, *b*) transforming the landmarks in the image to feature space, *c*) propagating the navigation state and feature space to the next time step, *d*) matching features from the propagated feature space to the new feature space, and, *e*) estimating the landmark location.

2.6.1 Image Collection. Digital imaging devices are designed to measure a pattern of light intensities projected through the optics and aperture onto a sensor. The pattern is a nonlinear function of the light intensity, optics, and pose of the device relative to the world. These devices provide a three-dimensional measurement of the world corrupted by measurement noise, optical distortions, and spatial aliasing. The three dimensions of the measurement include two spatial and one intensity dimension.

2.6.2 Transformation. Computers are not able to directly distinguish objects in an image, therefore, the image is transformed into feature vectors residing in feature space, using an algorithm to describe the pose and object dimensions of the feature. The algorithm chosen for the transformation is a variant of the scale-invariant feature tracking (SIFT) algorithm [9] as the stochastic projection can be applied to the pose with small effects to the object.

The feature transformation is processed by: *a*) computing the scale-space decomposition, *b*) detecting features, and *c*) calculating feature description vectors.

2.6.2.1 Scale-Space Decomposition. The scale-space decomposition is computed using a Gaussian spatial filter defined by the following function:

$$\mathbf{g}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.58}$$

where x and y are the spatial dimensions of the image in pixels and σ is the standard deviation of the blurring function.

The difference of a Gaussian filter is defined as:

$$\mathbf{f}(x, y, k, \sigma) = \mathbf{g}(x, y, k\sigma) - \mathbf{g}(x, y, \sigma) \tag{2.59}$$

where $k > 1$ is the scaling frequency step constant.

Given an initial standard deviation (σ_0), and a scaling frequency step constant (k) the ith difference of a Gaussian filter is:

$$\mathbf{f}(x, y, i) = \mathbf{g}(x, y, k^{i+1}\sigma_0) - \mathbf{g}(x, y, k^i\sigma_0) \tag{2.60}$$

Varying the scaling frequency step constant decomposes the image into multiple scale spaces centered on specific spatial frequencies. The step constant is varied in a manner to maintain equal spacing of the spatial frequencies.

When scale-space decomposition of the image is complete features may be detected readily.

2.6.2.2 Feature Detection. Candidate features are detected by locating local maxima or minima within the spatial and scale dimensions. Then the spatial detail for each candidate, centered on the candidate, is calculated as the eigenvalues of the matrix:

$$\mathbf{G} = \begin{bmatrix} \sum_{x,y\in\mathbf{W}} (\nabla\mathbf{f}_x)^2 & \sum_{x,y\in\mathbf{W}} \nabla\mathbf{f}_x \nabla\mathbf{f}_y \\ \sum_{x,y\in\mathbf{W}} \nabla\mathbf{f}_x \nabla\mathbf{f}_y & \sum_{x,y\in\mathbf{W}} (\nabla\mathbf{f}_x)^2 \end{bmatrix} \tag{2.61}$$

where $\mathbf{W}$ is the window over which the eigenvalues are computed, and $\nabla\mathbf{f}_x$ and $\nabla\mathbf{f}_y$ are the gradients of $\mathbf{f}(x, y, i)$ in the x and y directions.

The strongest candidates are chosen as features by thresholding the related eigenvalues and refining with the metric:

$$C(\mathbf{G}) = (1 + 2k_t)\sigma_1\sigma_2 + k_t(\sigma_1^2 + \sigma_2^2) \tag{2.62}$$

where σ_1 and σ_2 are the eigenvalues of $\mathbf{G}$, and k_t is a scaling parameter.

The pose subspace of each feature is constructed as:

$$\mathbf{z}_n^{pose}(t_i) = \begin{bmatrix} z_{n_x}(t_i) \\ z_{n_y}(t_i) \\ \sigma_n(t_i) \\ \theta_n(t_i) \end{bmatrix} \tag{2.63}$$

where z_{n_x}, z_{n_y}, σ_n and θ_n are the pixel location, scale and primary orientation of feature $n = 1...M$ for M features in the image at time t_i.

2.6.2.3 Feature Description. Finally, with the features within an image detected, the object dimensions (feature descriptions) are calculated as function of the intensity gradient of the scale-space surrounding each feature. The object dimensions of a feature are composed of the normalized histogram of the gradients around the feature.

When an image transformation is complete the image is represented by a collection of M vectors in feature space:

$$i(x, y, t_i) \rightarrow \mathbf{z}_n^*(t_i) \ \ \forall \, n \in \{1, ..., M\} \tag{2.64}$$

where

$$\mathbf{z}_n^*(t_i) = \begin{bmatrix} \mathbf{z}_n^{pose}(t_i)_{4\times1} \\ \mathbf{z}_n^{object}(t_i)_{128\times1} \end{bmatrix} \tag{2.65}$$

With the image transformation complete, the computed features are now propagated to the next time step.

2.6.3 Propagation. The state-space error estimate and landmark error are propagated from time t_i to t_{i+1} using the EKF developed in Section 2.5.2.

The landmarks detected within an image are assumed to be stationary. However, a small additive random walk is added to account for the effects of calibration and initialization errors, and to prevent the covariance from collapsing to zero. The landmark error is defined as:

$$\delta\dot{\mathbf{y}}^n(t) = \mathbf{G}_y(t)\mathbf{w}_y(t) \tag{2.66}$$

where $\mathbf{G}_y(t)$ is the influence matrix and $\mathbf{w}_y(t)$ is zero-mean, white Gaussian noise with covariance:

$$E\left\{\mathbf{w}(t)\mathbf{w}^T(t+\tau)\right\} = \mathbf{Q}(t)\delta(\tau) \tag{2.67}$$

The navigation and landmark error covariances are propagated with the following equations:

$$\begin{aligned} \mathbf{P}_{xx}(t_{i+1}^-) &= \mathbf{\Phi}(\mathbf{t_{i+1}}, \mathbf{t_i})\mathbf{P_{xx}}(\mathbf{t_i^+})\mathbf{\Phi^T}(\mathbf{t_{i+1}}, \mathbf{t_i}) \\ &\quad + \int_{t_i}^{t_{i+1}} \mathbf{\Phi}(t_{i+1}, \tau)\mathbf{G}(\tau)\mathbf{Q}(\tau)\mathbf{G}^T(\tau)\mathbf{\Phi}^T(t_{i+1}, \tau)d\tau && (2.68) \\ \mathbf{P}_{xy}(t_{i+1}^-) &= \mathbf{\Phi}(\mathbf{t_{i+1}}, \mathbf{t_i})\mathbf{P_{xy}}(\mathbf{t_i^+}) && (2.69) \\ \mathbf{P}_{yy}(t_{i+1}^-) &= \mathbf{P}_{yy}(t_i^+) + [t_{i+1} - t_i]\mathbf{G}_y(t_i)\mathbf{Q}_y(t_i)\mathbf{G}_y^T(t_i) && (2.70) \end{aligned}$$

2.6.4 Feature Matching. The next step in the vision-aiding algorithm is to statistically match the predicted features with a new set of measured features at time t_{i+1}. The match is performed by defining a metric, known as the Mahalanobis distance, to measure the quality of a match. The Mahalanobis distance is defined as the weighted inner product of the predicted feature vector, $\hat{\mathbf{z}}^*(t_{i+1})$, and the measured feature vector, $\mathbf{z}_n^*(t_{i+1})$:

$$D_n(t_{i+1}) = \left[\mathbf{z}_n^*(t_{i+1}) - \hat{\mathbf{z}}^*(t_{i+1})\right]^T \mathbf{P}_{z^*z^*}(t_{i+1}) \left[\mathbf{z}_n^*(t_{i+1}) - \hat{\mathbf{z}}^*(t_{i+1})\right] \tag{2.71}$$

where $\mathbf{P}_{z^*z^*}(t_{i+1})$ is the covariance of the predicted feature vector. Assuming scale and rotation independence of the pixel location, the covariance matrix can be written as:

$$\mathbf{P}_{z^*z^*}(t_{i+1}) = \begin{bmatrix} \mathbf{P}_{zz}(t_{i+1}) & 0 & 0 & 0 \\ 0 & \mathbf{P}_{\sigma\sigma} & 0 & 0 \\ 0 & 0 & \mathbf{P}_{\theta\theta} & 0 \\ 0 & 0 & 0 & \mathbf{P}_{z_dz_d} \end{bmatrix} \tag{2.72}$$

where $\mathbf{P}_{zz}$, $\mathbf{P}_{\sigma\sigma}$, $\mathbf{P}_{\theta\theta}$, and $\mathbf{P}_{z_dz_d}$ are the pixel location, scale, rotation, and descriptor uncertainties respectively.

As the SIFT algorithm provides no information regarding the statistical knowledge of $\mathbf{P}_{\sigma\sigma}$, $\mathbf{P}_{\theta\theta}$, and $\mathbf{P}_{z_dz_d}$, the scale and orientation are given zero weight and the distance metric is decomposed into pose and object descriptor distances:

$$D_{p_n}(t_{i+1}) = \left[\mathbf{z}_{p_n}(t_{i+1}) - \hat{\mathbf{z}}_p(t_{i+1})\right]^T \mathbf{P}_{z_pz_p}(t_{i+1}) \left[\mathbf{z}_{p_n}(t_{i+1}) - \hat{\mathbf{z}}_p(t_{i+1})\right] \tag{2.73}$$

$$D_{d_n}(t_{i+1}) = \left[\mathbf{z}_{d_n}(t_{i+1}) - \hat{\mathbf{z}}_d(t_{i+1})\right]^T \mathbf{P}_{z_dz_d}(t_{i+1}) \left[\mathbf{z}_{d_n}(t_{i+1}) - \hat{\mathbf{z}}_d(t_{i+1})\right] \tag{2.74}$$

where

$$\mathbf{P}_{z_pz_p}(t_{i+1}) = \begin{bmatrix} \mathbf{P}_{zz}(t_{i+1}) & 0 & 0 & 0 \\ 0 & \infty & 0 & 0 \\ 0 & 0 & \infty & 0 \end{bmatrix} \tag{2.75}$$

$$\mathbf{P}_{z_dz_d}(t_{i+1}) = \mathbf{I} \tag{2.76}$$

The result of decomposing the distance is to *a*) include only the effects of the predicted pixel location in the pose distance, and *b*) weight all of the components of the object descriptor equally.

The two distance metrics are used in succession to improve the search speed and maintain pixel error measurement independence. A successful match is found when the object distance is below a pre-defined threshold. Additionally, the application of a uniqueness filter is available to reject insufficiently distinct feature matches.

2.6.5 Landmark Location Estimation.

Landmark location estimates are computed using using binocular stereopsis as it does not require a terrain model and is therefore appropriate for navigation in unknown environments. For convenience, a binocular reference frame c_0 is defined midway between the two camera frames c_a and c_b.

The computation of landmark locations is completed in three steps: *1*) feature selection and matching *2*) calculation of line of sight from the binocular origin to the landmark, and *3*) estimating the landmark location.

2.6.5.1 Feature Selection and Matching.

The selection of a feature is performed on one of the images of the binocular pair as described in Section 2.6.2.2.

The resulting feature is then statistically projected into the feature space of the second image. For example, given a feature in the c_a-*frame* the projected pixel location in the c_b-*frame* is computed as:

$$\mathbf{z}^b = \mathbf{T}_{c_b}^{pix}\underline{\mathbf{s}}^{c_b} \tag{2.77}$$

$$\mathbf{s}^{c_b} = \mathbf{C}_{c_0}^{c_b}\left[k\mathbf{C}_{c_a}^{c_0}\mathbf{T}_{pix}^{c_a}\mathbf{z}^a + \mathbf{p}_{cam_a}^{c_0} - \mathbf{p}_{cam_b}^{c_0}\right] \tag{2.78}$$

where $\mathbf{T}_{pix}^{c_a}$ and $\mathbf{T}_{c_b}^{pix}$ are the camera projection matrices for cameras a and b, $\mathbf{p}_{cam_a}^{c_0}$ and $\mathbf{p}_{cam_b}^{c_0}$ are the location vectors of the camera frames relative to the binocular frame, and k is the unknown distance parameter.

Variations in the distance parameter describe an epipolar line in the c_b-*frame* image modeled as a Gaussian distribution with mean and variance:

$$\bar{k} = E[k] = 4.5\ d_{binoc} \tag{2.79}$$

$$\sigma_k^2 = E\left[(k - \bar{k})^2\right] = 9\ d_{binoc}^2 \tag{2.80}$$

where the binocular disparity d_{binoc} is defined as:

$$d_{binoc} = ||\mathbf{p}_{cam_a}^{c_0} - \mathbf{p}_{cam_b}^{c_0}|| \tag{2.81}$$

Matching of features is performed by determining the features within a statistical distance of the prediction and choosing the feature with the smallest feature description distance. The method for matching features is the same a discussed in Section 2.6.4.

2.6.5.2 Line of Sight. Next, the estimate of the relative line of sight from the c_0-*frame* origin to the landmark is computed from the pixel locations of the landmark in each camera frame:

$$\mathbf{z}^a = \mathbf{T}_{c_a}^{pix}\mathbf{C}_{c_0}^{c_a}\left[\mathbf{s}_0^{c_0} - \mathbf{p}_{cam_a}^{c_0}\right] \tag{2.82}$$

$$\mathbf{z}^b = \mathbf{T}_{c_b}^{pix}\mathbf{C}_{c_0}^{c_b}\left[\mathbf{s}_0^{c_0} - \mathbf{p}_{cam_b}^{c_0}\right] \tag{2.83}$$

The estimation results in an estimated line of sight vector, $\hat{\mathbf{s}}_0^{c_0}$, and corresponding covariance, $\mathbf{P}_{s_o s_0}$.

2.6.5.3 Location Estimation. Finally, the *n-frame* landmark location is estimated by substituting the line of sight vector estimate and navigation state estimate into the landmark location equation:

$$\mathbf{y}^n = \mathbf{p}^n + \mathbf{C}_b^n\left[\mathbf{p}_0^b + \mathbf{C}_{c_0}^b\mathbf{s}_0^{c_0}\right] \tag{2.84}$$

In the next section, the concepts of federated filtering are discussed, providing the architecture upon which the sharing of information, and navigation state estimates are developed.

2.7 Federated Filtering

The federated filter, as described in [2], is an information-sharing filter, therefore, before the federated filter may be described the term *information* must be defined. The following definition of *information*, from [2], is used throughout this thesis.

> *Information within an optimal estimator (filter) generally represents knowledge of the random process (state) being estimated. Information is gained from initial state estimates and periodic measurements. Information is lost through propagation uncertainties (processed noise). Specifically, information within a filter is represented by its information matrix and its information state vector (or equivalently, by its error covariance matrix and standard state vector). The information matrix (covariance matrix inverse) is a statistical measure of the amount of information present. The information state vector (product of the information matrix and standard state vector) represents specific knowledge of the actual random process. Greater information implies smaller estimation errors, and vice versa.*

The federated filter is designed to share information amongst multiple local filters and one master filter, and from their solutions build a total solution. The basic information sharing methodology implemented by the federated filter is:

- divide the total system information among several component local filters and a single master filter;
- perform local and master filter time propagation and measurement processing, adding local sensor information;
- recombine the updated local and master filter information in to a new total sum.

A typical federated filter set up is depicted in Figure 2.2. Local sensors, 1...n, each have an associated local filter with inputs from the local sensor and a common

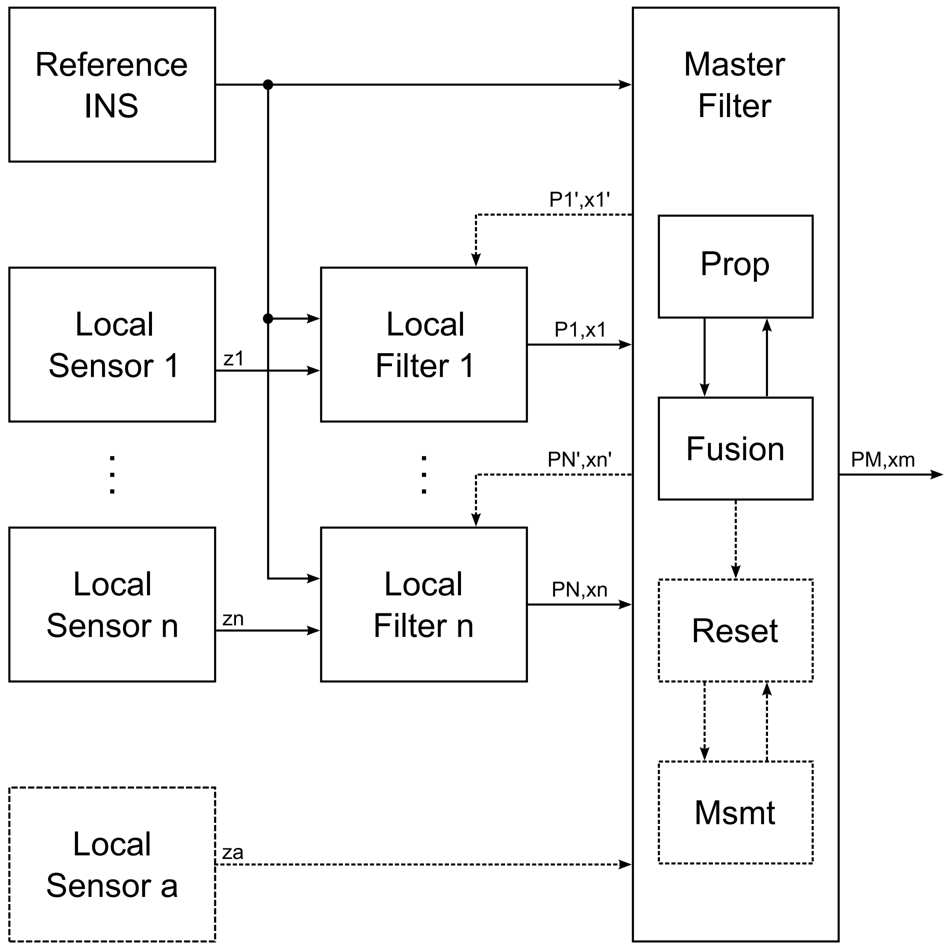


Figure 2.2: Information Partitioning in Federated Filter [2].

reference INS. Additional local sensor measurements without a local filter are provided directly to the master filter along with the solutions of the local filters.

The equivalent centralized filter solution is represented by the covariance matrix $\mathbf{P}$ and state vector $\hat{\mathbf{x}}$. Similarly, the local filter solutions are represented by $\mathbf{P}l$ and $\hat{\mathbf{x}}l$; and the master filter by $\mathbf{P}m$ and $\hat{\mathbf{x}}m$. The index $i = 1 \ldots n$ is used to index the local filters alone, while the index $k = 1 \ldots n, m$ is used to index the local filters and a master filter, where $k = m$ represents the master filter.

2.7.1 Construction. The formulation in this section is derived from [2]. The federated filter is constructed such that statistically independent local and master filter solutions can be combined at any time with the following additive information algorithm,

$$\mathbf{P}^{-1} = \mathbf{P}m^{-1} + \mathbf{P}1^{-1} + \ldots + \mathbf{P}n^{-1} \tag{2.85}$$

$$\mathbf{P}^{-1}\hat{\mathbf{x}} = \mathbf{P}m^{-1}\hat{\mathbf{x}}m + \mathbf{P}1^{-1}\hat{\mathbf{x}}1 + \ldots + \mathbf{P}n^{-1}\hat{\mathbf{x}}n \tag{2.86}$$

where the inverse covariance matrix ($\mathbf{P}^{-1}$) is known as the "information matrix".

Proper construction of the federated filter is accomplished by applying a fraction of the total information to each local filter and the master filter,

$$\mathbf{P}k^{-1} = \mathbf{P}^{-1}\,\beta_k \;\text{ or }\; \mathbf{P} = \mathbf{P}\,\beta_k^{-1} \tag{2.87}$$

while maintaining constant total information across the sum to satisfy the *conservation of information principle*. Therefore, the share-fraction values (β_k) must sum to unity:

$$\sum_{k=1}^{n,m} \beta_k = \beta_m + \sum_{i=1}^{n} \beta_i = 1 \tag{2.88}$$

The share-fraction values (β_k) are only applied to the information, therefore, the local and master filter state vectors are equal to the total state vector.

$$\hat{\mathbf{x}}k = \hat{\mathbf{x}} \tag{2.89}$$

Application of the local and master filter solutions given by (2.87) and (2.89) to the additive information algorithm (2.85) demonstrates the local and master filter solutions can be combined to yield the correct total solution. $(\mathbf{P},\ \hat{\mathbf{x}})$.

$$\mathbf{P}^{-1} = \mathbf{P}m^{-1} + \mathbf{P}1^{-1} + \ldots + \mathbf{P}n^{-1} \tag{2.90}$$

$$= \mathbf{P}^{-1}\ \beta_m + \mathbf{P}^{-1}\ \beta_1 + \ldots + \mathbf{P}^{-1}\ \beta_n \tag{2.91}$$

$$= \mathbf{P}^{-1} \sum_{k=1}^{n,m} \beta_k \tag{2.92}$$

$$= \mathbf{P}^{-1} \tag{2.93}$$

$$\mathbf{P}^{-1}\hat{\mathbf{x}} = \mathbf{P}m^{-1}\hat{\mathbf{x}}m + \mathbf{P}1^{-1}\hat{\mathbf{x}}1 + \ldots + \mathbf{P}n^{-1}\hat{\mathbf{x}}n \tag{2.94}$$

$$= \mathbf{P}^{-1}\ \beta_m\hat{\mathbf{x}} + \mathbf{P}^{-1}\ \beta_1\hat{\mathbf{x}} + \ldots + \mathbf{P}^{-1}\ \beta_n\hat{\mathbf{x}} \tag{2.95}$$

$$= \mathbf{P}^{-1}\hat{\mathbf{x}} \sum_{k=1}^{n,m} \beta_k \tag{2.96}$$

$$= \mathbf{P}^{-1}\hat{\mathbf{x}} \tag{2.97}$$

When the common process noise information is divided in the same fashion, the discrete time propagation process can be performed by independent, parallel operations of the local and master filters.

2.7.2 Propagation. The local and master filters are propagated, in parallel, provided the common process noise information is divided in the same fashion as the information matrix. The covariance propagation equations from time $t-1$ to t are:

$$\mathbf{P}k_t = \mathbf{\Phi}k_t\ \mathbf{P}k_{(t-1)}\ \mathbf{\Phi}k_t^T + \mathbf{G}k_t\ \mathbf{Q}k_t\ \mathbf{G}k_t^T \tag{2.98}$$

Assuming the local and master filters are full-sized, the transition matrices $\mathbf{\Phi}k$ equal $\mathbf{\Phi}$, and the noise distribution matrices $\mathbf{G}k$ equal $\mathbf{G}$. However, the process noise covariance matrices $\mathbf{Q}k$ are governed by the information-sharing principle.

$$\mathbf{Q}k^{-1} = \mathbf{Q}^{-1}\ \beta_k \ \text{ or } \ \mathbf{Q} = \mathbf{Q}\ \beta_k^{-1} \tag{2.99}$$

The post-propagation information matrix can be obtained through the additive information algorithm, when the information matrices are obtained through (2.87), and the noise information matrices are obtained through (2.99).

$$\begin{aligned}
\sum_{k=1}^{n,m} \mathbf{P}k^{-1} &= \sum_{k} \left[\mathbf{\Phi}_t \, \mathbf{P}_{(t-1)} \, \beta k^{-1} \, \mathbf{\Phi}_t^T + \mathbf{G}_t \, \mathbf{Q}_t \, \beta k^{-1} \, \mathbf{G}_t^T\right]^{-1} && (2.100)\\
&= \left[\sum_{k=1}^{n,m} \beta k\right] \left[\mathbf{\Phi}_t \, \mathbf{P}_{(t-1)} \, \mathbf{\Phi}_t^T + \mathbf{G}_t \, \mathbf{Q}_t \, \mathbf{G}_t^T\right]^{-1} && (2.101)\\
&= \mathbf{P}^{-1} && (2.102)
\end{aligned}$$

2.7.3 Measurement Update. After propagation each local filter incorporates discrete measurements $\tilde{\mathbf{z}}i$ from its sensor. Measurement information is added to local filter i through,

$$\begin{aligned}
\mathbf{P}i_+^{-1} &= \mathbf{P}i^{-1} + \mathbf{H}i \, \mathbf{R}i^{-1} \, \mathbf{H}i^T && (2.103)\\
\mathbf{P}i_+^{-1} \, \hat{\mathbf{x}}i_+ &= \mathbf{P}i^{-1} \, \hat{\mathbf{x}}i + \mathbf{H}i \, \mathbf{R}i^{-1} \, \tilde{\mathbf{z}}i && (2.104)
\end{aligned}$$

where the subscript + refers to posts measurement values, $\mathbf{R}i^{-1}$ is the local filter measurement information matrix and $\mathbf{H}i = (\delta \mathbf{z}i/\delta \mathbf{x}i)^T$.

Using the additive information algorithm to combine the results of local filter measurement updates provides the correct total solution,

$$\begin{aligned}
\mathbf{P}m^{-1} + \sum_{i=1}^{n} &\left[\mathbf{P}i^{-1} + \mathbf{H}i \, \mathbf{R}i^{-1} \, \mathbf{H}i^T\right] && (2.105)\\
&= \mathbf{P}^{-1} + \sum_{i=1}^{n} \mathbf{H}i \, \mathbf{R}i^{-1} \, \mathbf{H}i^T && (2.106)\\
&= \mathbf{P}_+^{-1} && (2.107)
\end{aligned}$$

and,

$$\mathbf{P}m^{-1}\,\hat{\mathbf{x}}m + \sum_{i=1}^{n}\left[\mathbf{P}i^{-1}\,\hat{\mathbf{x}}i + \mathbf{H}i\,\mathbf{R}i^{-1}\,\tilde{\mathbf{z}}i\right] \tag{2.108}$$

$$= \mathbf{P}^{-1}\,\hat{\mathbf{x}} + \sum_{i=1}^{n}\mathbf{H}i\,\mathbf{R}i^{-1}\,\tilde{\mathbf{z}}i \tag{2.109}$$

$$= \mathbf{P}_{+}^{-1}\,\hat{\mathbf{x}}_{+} \tag{2.110}$$

Thus the federated filter provides a solution which would be achieved by a single filter processing all of the sensor measurement sets. However, the federated filter implementation of information sharing provides three advantages over a single filter implementation [2]:

- increased measurement data throughput by parallel operation of local filters, and by data compression within local filters;
- enhanced system fault-tolerance by maintaining multiple component solutions to improve fault detection and recovery capabilities; and
- improved accuracy and stability of cascaded filter operations, via use of theoretically correct estimation algorithms.

2.7.4 Filter Reset. Finally, the federated filter may be implemented using one of four different local filter reset architectures [3]. The four architectures, also know as reset modes, are *1) no-reset*, *2) fusion-reset*, *3) zero-reset*, and *4) rescale.*

In the *no-reset* mode, the local filters operate autonomously while maintaining the long-term information of the navigation system. The master filter fuses the local filter solutions at each time step and propagates the fused solution to the next time step. The master filter discards the propagated solution prior to fusing the next set of local filter solutions.

The local filters retain the long-term system information in the *fusion-reset* mode, but do not operate autonomously as in the *no-reset* mode. In this mode the master filter divides the fused information and applies an equal share to each of the local filters. Each local filter begins the next cycle with the same information as the other filters. Therefore, the weaker local filters gain information from the stronger filters, which may help constrain the weaker filters.

In the *zero-reset* mode the long-term system information storage is transfered to the master filter. Each local filter acts as a integrate-and-dump filter compressing it's sensor measurements into a single solution. After the local filters send their solutions to the master filter, they reset themselves to zero information. This ensures the next set of inputs contain no old information.

The *rescale* mode is similar to the *zero-reset* mode, except the local filters send half of their information to the master filter. Additionally, the local filters reset their information to one-half instead of zero after sending the solution. This mode allows the local filters to retain enough information to make the filter locally usable in the event of failure.

2.8 Multiple Vehicles

In this section, two approaches to cooperative multiple platform navigation are discussed and analyzed. Additional approaches to multiple platform navigation may be found in [1, 8, 11, 16, 17].

2.8.1 Sanderson. In [18] Sanderson proposes a CNS based on the application of inter-platform range measurements to a linear Kalman filter. The use of such range measurements requires each platform to estimate the position of all other platforms to which it performs a range measurement. This poses a problem during the propagation step of the Kalman filter as each platform does not have implicit knowledge of the trajectory of the other platforms. Sanderson mitigates this problem by pre-programming each platform with a trajectory consisting of a series of desire positions. At each time step the platforms transmit their next desired position so the other platforms may properly propagate the position estimate.

Sanderson demonstrates the the validity of the CNS with three, 10 run, experiments using two robots traveling in parallel along a $10\,m$ long straight path. In the first experiment to two robots do not cooperate, resulting in an average error of approximately $10\,cm$ at the end of the $10\,m$ run. The second experiment is performed with the robots cooperating, resulting in an average error less than $10\,cm$ at the end of the run. In both of these cases the position estimate is diverging from truth at the end of the run. Finally, the third experiment uses the CNS with the addition of one of the robots following a wall at a set distance. This results in a less than 1cm average error for the robot which is not following the wall.

The CNS developed by Sanderson was created for the navigation of platforms constrained to a two-dimensional surface. However, the concepts used in the CNS development may be extrapolated to three dimensions. A major limiting factor in the use of this system for large numbers of platforms is the requirement for each platform to estimate the other platforms positions. This requirement increases the number of states in the Kalman filter and will therefore increase the processing time of the system with increased numbers of platforms.

2.8.2 Panzieri, Pascucci and Setola. In [15] Panzieri, Pascucci and Setola circumvent the problems with large state vectors, due to the number of platforms in the network, by implementing a CNS using an interlaced Kalman filter. Using this method each platform only estimates its own pose and shares estimated positions and covariances when in communication range of other platforms. Additionally, when platforms are close enough, range measurements may be incorporated into the solution.

The CNS is demonstrated using five `Matlab`® simulations. The simulations consist of 10 platforms operating within a 12×12 arena which includes nine landmark beacons. The five simulations progress from a complete non-cooperative system to a full system capable of communication between platforms and all of the landmark beacons. Over the simulations the mean position error of the 10 platforms is reduced from $2\,m$, for the non-cooperative system, to less than $2\,cm$ for the full CNS.

The CNS system developed by Panzieri, Pascucci and Setola demonstrates the ability of a interlaced Kalman filter approach to reduce the state vector of a platform, increasing the rate at which the solution is computed.

In both approaches to the CNS developed utilize only information pertaining to the condition of the platforms. In the next chapter, vision-aiding of an INS is extended to use with multiple platforms cooperating to improve the navigation solution of the group.

III. Methodology

This chapter fully develops two methods for the implementation of the multiple vehicle, cooperative, vision-aided navigation system. The chapter begins with an overview of the multiple platform federated filter and describes the differences between the two methods of implementation. Next, a new approach of local filter scaling is discussed to simplify the development of the primary local filter implemented in the first method. Then, the primary local filter is defined followed by descriptions of the landmark extraction and matching algorithms. Next, the range measurement algorithm is developed followed by the formation of the secondary local filter function. Finally, the fusion algorithm which comprises the master filter is developed.

3.1 Multiple Platform Federated Filter

The federated filter as developed in Section 2.7 is designed for the express purpose of fusing navigation solutions from multiple local-level filters on a single platform utilizing multiple sensor measurements. The federated filters developed for this research operate in the same manner, differing only in the source of sensor measurements for the secondary local filter.

The first method of implementation, designated the "post-scaling" method and shown in Figure 3.1, is developed to utilize the algorithms developed by Veth, as detailed in Section 2.6, with minimal modification. The objective of this method is to implement these algorithms in the primary local filter with no modifications. As the covariance matrix is pre-scaled in a federated filter, this method requires the development of a new way to scale the covariance matrix, after the propagation and measurement update of the Kalman filter.

The second method of implementation, designated the "pre-scaling" method and shown in Figure 3.2, modifies the algorithms developed by Veth. These modifications allow for the pre-scaling of the covariance matrix, and operation of the filter for a single time step.

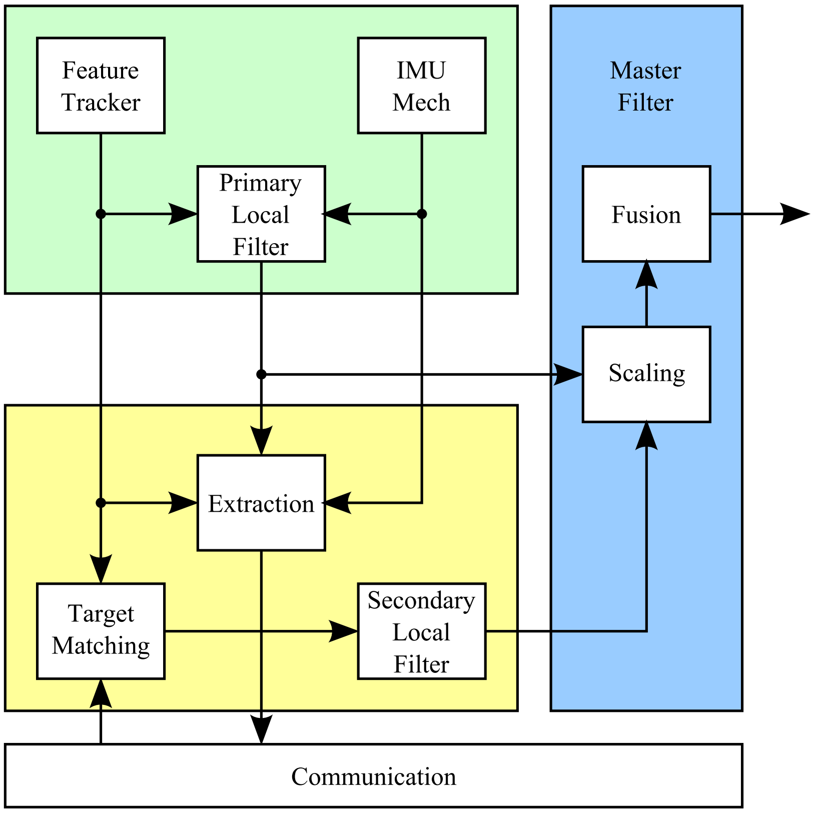


Figure 3.1: Platform Block Diagram (Method 1): The primary local filter (green) receives measurements from on-board sensors. The secondary local local filter (yellow) receives measurements from remote platforms through the extraction and matching functions. The master filter (blue) scales the outputs and fuses the results of the two local filters to provide a total system solution.

The basic structure of the navigation system is the same regardless of the method of implementation. The system consists of a primary and secondary local filter, a landmark extraction algorithm, a landmark matching algorithm, and a master filter.

The most significant difference between the two methods of implementation is when the local filter solutions are scaled. The post-scaling method, shown in Figure 3.1, performs local filter scaling in the master filter prior to fusion of the local solutions. In Figure 3.2 the scaling block is removed as the pre-scaling method scales the local filters during the initialization of the primary filter.

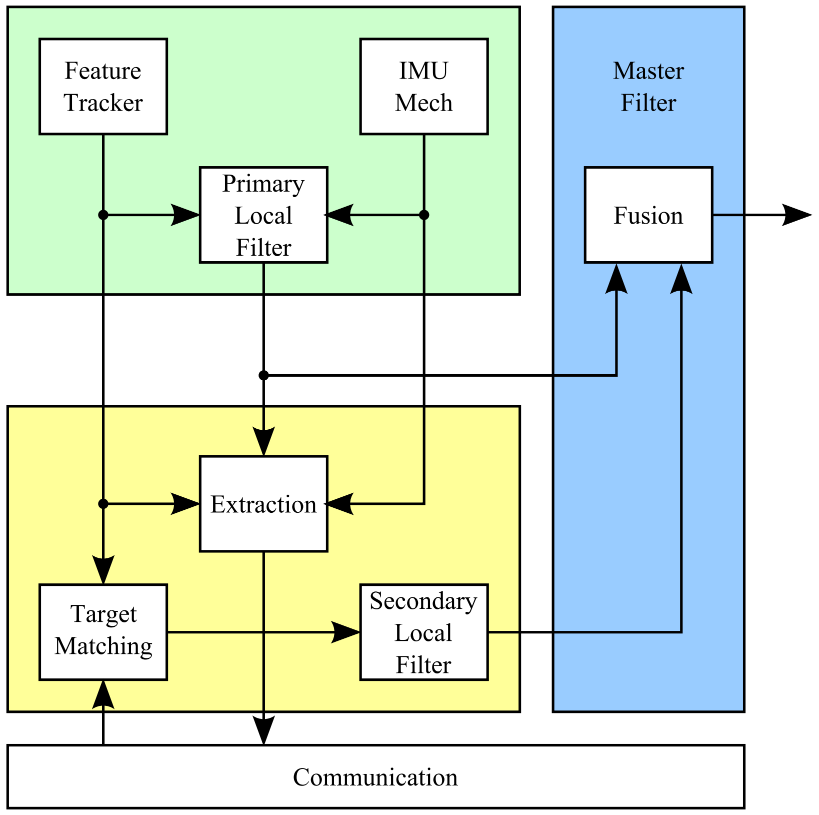


Figure 3.2: Platform Block Diagram (Method 2): The primary local filter (green) receives measurements from on-board sensors. The secondary local local filter (yellow) receives measurements from remote platforms through the extraction and matching functions. The master filter (blue) fuse the results of the two local filters to provide a total system solution.

In the next section, the new local scaling method is developed for the implementation of the post-scaling navigation system method.

3.2 Local Filter Scaling

To utilize the navigation filter developed by Veth, without major modifications to scale the covariance matrices, a new scaling method is developed. The formation of this new scaling method begins with the definitions of the local covariance matrices, at any time t, provided in [3],

$$\mathbf{P}l_t = \begin{bmatrix} \mathbf{P}C_t\ \gamma_{pcl} & 0 \\ 0 & \mathbf{P}B_t\ \gamma_{pbl} \end{bmatrix} \tag{3.1a}$$

$$\mathbf{Q}l_t = \begin{bmatrix} \mathbf{Q}C_t\ \gamma_{pcl} & 0 \\ 0 & \mathbf{Q}B_t\ \gamma_{pbl} \end{bmatrix} \tag{3.1b}$$

$$\mathbf{R}l_t = \begin{bmatrix} \mathbf{R}C_t\ \gamma_{pcl} & 0 \\ 0 & \mathbf{R}B_t\ \gamma_{pbl} \end{bmatrix} \tag{3.1c}$$

where ($\mathbf{P}l_t$, $\mathbf{Q}l_t$, $\mathbf{R}l_t$) are the covariance matrices of the local filter, ($\mathbf{P}C_t$, $\mathbf{Q}C_t$, $\mathbf{R}C_t$) are the portions of the full system covariance matrices common to all of the local filters, ($\mathbf{P}B_t$, $\mathbf{Q}B_t$, $\mathbf{R}B_t$) are the portions of the full system covariance matrices unique to the local filter, and (γ_{pcl}, γ_{pbl}) are the federated filter, covariance mode, scaling parameters.

The full state estimate vector is divided into two categories: common states, and unique states. Common states are those states estimated by all local filters, while unique states are estimated only by the local filter which performs a measurement of the state.

The primary and secondary local filters in Figure 3.1 share all of the system states. With no unique states in either of the local filters, the scaling parameters are equal, and $\gamma_{pl} = \gamma_{pcl} = \gamma_{pbl}$. The scaling parameter (γ_{pl}) is extracted from the matrices and the local filter covariances become:

$$\mathbf{P}l_t = \gamma_{pl}\ \mathbf{P}_t \tag{3.2a}$$

$$\mathbf{Q}l_t = \gamma_{pl}\ \mathbf{Q}_t \tag{3.2b}$$

$$\mathbf{R}l_t = \gamma_{pl}\ \mathbf{R}_t \tag{3.2c}$$

The two local filters share all of the states for which measurements are available, therefore, the measurement noise covariance ($\mathbf{R}_t$) must be scaled as well as the process ($\mathbf{P}_t$) and process noise ($\mathbf{Q}_t$) covariances.

The local filter state vector and process covariance are initialized, in covariance mode, as defined in [3].

$$\mathbf{x}l_t^+ = \mathbf{x}_t^+ \tag{3.3}$$

$$\mathbf{P}l_t^+ = \gamma_{pl}\,\mathbf{P}_t^+ \tag{3.4}$$

where t is equal to zero for initialization.

As shown in [2] the state and noise transformation matrices do not require scaling.

$$\mathbf{\Phi}l_t = \mathbf{\Phi}_t \tag{3.5}$$

$$\mathbf{G}l_t = \mathbf{G}_t \tag{3.6}$$

The states are propagated as follows:

$$\mathbf{x}l_t^- = \mathbf{\Phi}l_t\,\mathbf{x}l_{(t-1)}^+ \tag{3.7}$$

Substituting (3.3) and (3.5) into (3.7) then solving for $\mathbf{x}l_t^-$ provides,

$$\mathbf{x}l_t^- = \mathbf{\Phi}_t\,\mathbf{x}_{(t-1)}^+ \tag{3.8}$$

$$= \mathbf{x}_t^- \tag{3.9}$$

Next, the covariance matrix is propagated,

$$\mathbf{P}l_t^- = \mathbf{\Phi}l_t\,\mathbf{P}l_{(t-1)}^+\,\mathbf{\Phi}l_t^T + \mathbf{G}l_t\,\mathbf{Q}l_t\,\mathbf{G}l_t^T \tag{3.10}$$

Substituting (3.2a), (3.2b) and (3.6) into (3.10) then solving for $\mathbf{P}l_t^-$ provides,

$$\begin{aligned}
\mathbf{P}l_t^- &= \mathbf{\Phi}_t\ \gamma_{pl}\ \mathbf{P}^+_{(t-1)}\ \mathbf{\Phi}_t^T + \mathbf{G}_t\ \gamma_{pl}\ \mathbf{Q}_t\ \mathbf{G}_t^T && (3.11)\\
&= \gamma_{pl}\ (\mathbf{\Phi}_t\ \mathbf{P}^+_{(t-1)}\ \mathbf{\Phi}_t^T + \mathbf{G}_t\ \mathbf{Q}_t\ \mathbf{G}_t^T) && (3.12)\\
&= \gamma_{pl}\ \mathbf{P}_t^- && (3.13)
\end{aligned}$$

Therefore, the state vector and process covariance may be propagated from one time step to the next, and the scaling factor applied after the propagation is complete. However, the new scaling method must continue through the measurement update process to be useful.

As demonstrated in [2] the measurement transformation matrix is not scaled.

$$\mathbf{H}l_t = \mathbf{H}_t \tag{3.14}$$

With the measurement transformation matrix (3.14) and process covariance matrix (3.11) defined, the Kalman gain is computed,

$$\mathbf{K}l_t = \mathbf{P}l_t^-\ \mathbf{H}l_t^T\ \left(\mathbf{H}l_t\ \mathbf{P}l_t^-\ \mathbf{H}l_t^T + \mathbf{R}l_t\right)^{-1} \tag{3.15}$$

Substituting (3.2c), (3.11) and (3.14) into (3.15) then solving for $\mathbf{K}l_t$ provides,

$$\begin{aligned}
\mathbf{K}l_t &= \gamma_{pl}\ \mathbf{P}_t^-\ \mathbf{H}_t^T\ \left(\mathbf{H}_t\ \gamma_{pl}\ \mathbf{P}_t^-\ \mathbf{H}_t^T + \gamma_{pl}\ \mathbf{R}_t\right)^{-1} && (3.16)\\
&= \gamma_{pl}\ \mathbf{P}_t^-\ \mathbf{H}_t^T \frac{1}{\gamma_{pl}} \left(\mathbf{H}_t\ \mathbf{P}_t^-\ \mathbf{H}_t^T + \mathbf{R}_t\right)^{-1} && (3.17)\\
&= \mathbf{P}_t^-\ \mathbf{H}_t^T\ \left(\mathbf{H}_t\ \mathbf{P}_t^-\ \mathbf{H}_t k^T + \mathbf{R}_t\right)^{-1} && (3.18)\\
&= \mathbf{K}_t && (3.19)
\end{aligned}$$

This computation of the Kalman gain is only valid if the measurement noise covariance ($\mathbf{R}_t$) is scaled identically to the process ($\mathbf{P}_t$) and process noise ($\mathbf{Q}_t$) covariances. Otherwise, the scaling parameters will not cancel and the local filter Kalman gain requires pre-scaling.

As with the state vector and measurement transformation matrix, the measurement vector does not require scaling.

$$\mathbf{z}l_t = \mathbf{z}_t \tag{3.20}$$

The measurement update of the state is:

$$\mathbf{x}l_t^+ = \mathbf{x}l_t^- + \mathbf{K}l_t \left(\tilde{\mathbf{z}}l_t - \mathbf{H}l_t^T \, \mathbf{x}l_t^-\right) \tag{3.21}$$

Substituting (3.8), (3.16), (3.20) and (3.14) into (3.21) then solving for $\mathbf{x}l_t^+$ provides,

$$\begin{align} \mathbf{x}l_t^+ &= \mathbf{x}_t^- + \mathbf{K}_t \left(\tilde{\mathbf{z}}_t - \mathbf{H}_t^T \, \mathbf{x}_t^-\right) \tag{3.22} \\ &= \mathbf{x}_t^+ \tag{3.23} \end{align}$$

Finally, the process covariance is updated,

$$\mathbf{P}l_t^+ = \mathbf{P}l_t^- - \mathbf{K}l_t \, \mathbf{H}l_t^T \, \mathbf{P}l_t^- \tag{3.24}$$

Substituting (3.11), (3.16), and (3.14) into (3.24) then solving for $\mathbf{P}l_t^+$ provides,

$$\begin{align} \mathbf{P}l_t^+ &= \gamma_{pl} \, \mathbf{P}_t^- - \mathbf{K}_t \, \mathbf{H}_t^T \, \gamma_{pl} \, \mathbf{P}_t^- \tag{3.25} \\ &= \gamma_{pl} \left(\mathbf{P}_t^- - \mathbf{K}_t \, \mathbf{H}_t^T \, \mathbf{P}_t^-\right) \tag{3.26} \\ &= \gamma_{pl} \, \mathbf{P}_t^+ \tag{3.27} \end{align}$$

Therefore, the measurement update of the state vector and process covariance does not require pre-scaling of the process covariance. This result allows for a set of local filters to run independently without scaling the information applied to them. Samples are taken from the local filters as needed and scaled prior to fusion by the master filter.

3.3 Primary Local Filter

The primary local filter implements the navigation system developed by Veth, as described in Section 2.6. Measurements to the filter are provided by the onboard vision system.

3.3.1 Post-Scaling. The only modification to the algorithm, to implement the post-scaling method, is the addition of a single function to extract the initial covariance related to a landmark when a new landmark is added to the navigation state. This function is applied to both methods and is detailed in Section 3.3.3. The resulting state and process covariance of the primary filter are scaled in the master filter prior to fusion.

3.3.2 Pre-Scaling. The pre-scaling method makes three modifications to the original Veth algorithm. First, the initial covariance related to a landmark is extracted in the same manner as in the post-scaling method. Next, the process (P) and processed noise (Q) covariances are scaled by the scaling factor (γ). The scaling factor is 2 when the navigation system is run in no-reset mode and 3 in fusion-reset mode, as described in Section 2.7.4.

The final modification to the Veth implementation is the removal of time looping. The original code processes the entire simulation in one call of the code and then provides results for every time step. The modification removes the loop to provide a result for a single time step. This modification allows for the pre-scaling of the covariances and operation of the navigation system in one of the reset modes.

3.3.3 Initial Covariance Extraction. Immediately after a landmark is added to the primary filter, the initial covariance extraction function creates and saves two matrices. The first matrix ($\mathbf{M}$) is a mask of the process covariance matrix and consists of zeros in the three columns and rows pertaining to the added landmark, and ones elsewhere. The second matrix ($\mathbf{L}$) is a copy of the process covariance matrix with all values outside of the three columns and rows pertaining to the added landmark set to zero. These two matrices are computed once for every landmark added to the primary filter, and are provided to the secondary filter to facilitate resetting the process covariance when a new landmark match is made.

The output of the primary filter at each time step is now applied to the master filter and landmark extraction function.

3.4 Landmark Extraction

The landmark extraction function computes the error corrected n-frame position of the platform and all landmarks tracked by the primary filter. The function then extracts the platform position, landmark positions, and all related variances. Additionally, the token, scale, and rotation data for each landmark is extracted to facilitate landmark matching.

The first step in the function is to correct the position of the platform and landmarks by subtracting the pertinent error state from the estimated platform and landmark position estimates:

$$\mathbf{p}^n = \hat{\mathbf{p}}^n - \tilde{\mathbf{p}}^n \tag{3.28}$$

$$\mathbf{y}_k^n = \hat{\mathbf{y}}_k^n - \tilde{\mathbf{y}}_k^n \tag{3.29}$$

where $\mathbf{p}^n$ is the corrected platform position, and $\mathbf{y}_k^n$ is the corrected landmark position for the k^{th} landmark.

Next the covariance of the platform position is extracted from the full navigation state covariance matrix.

$$\mathbf{R}_{pp} = \mathbf{P}_{pp} - \mathbf{P}_{px}\mathbf{P}_{xx}\mathbf{P}_{px}^T \tag{3.30}$$

where $\mathbf{R}_{pp}$ is the resulting covariance, $\mathbf{P}_{pp}$ is the portion of the state covariance related to the platform position, $\mathbf{P}_{xx}$ is the portion of the state covariance related to all other states, and $\mathbf{P}_{px}$ is the portion of the state covariance relating the position to all of the other states. This result provides the platform position covariance including the effects of all other states and their related measurements. The platform position and covariance are extracted for use in the range measurements as detailed in Section 3.6.

Finally, the variances of the landmarks are extracted. The extraction function used for the platform position is not valid for the landmarks as it will include the information about the platform position. This information will cause a bias when the result is applied to a second platform. Therefore, the variance for each landmark is:

$$\mathbf{R}_{y_k y_k} = diag(\mathbf{P}_{y_k y_k}) \tag{3.31}$$

where $k = 1 \ldots n$ is the k^{th} landmark, $\mathbf{R}_{y_k y_k}$ is the resulting variance, and $\mathbf{P}_{y_k y_k}$ is the 3×3 portion of the process covariance related to the landmark. Extraction of the landmark variances by this method is not optimal as information is lost in the extraction. However, if the information is extracted in an optimal manner and shared with other platforms, a bias is induced in the position estimate due to the difference in position of the platforms.

After all required data is extracted from the primary filter the result is transferred, through communications channels, to all other platforms participating in the cooperative system.

The next section details the target matching function using the extracted data from other platforms.

3.5 Target Matching

The landmark matching function provides the secondary local filter with measurements of landmark positions provided by other platforms connected to the cooperative navigation system. The landmarks of interest are limited to those tracked by the primary local filter. Therefore, this function compares the landmarks sent from other platforms to those tracked by the primary filter and discards any which do not match.

A match correlation value (A_{jk}) is computed for every possible combination of primary filter landmark (j) and cooperative platform landmark (k). This value is computed as the arc cosine of the dot product of the SIFT tokens for each primary filter and cooperative platform landmark combination:

$$A_{jk} = \arccos(T_j \cdot T_k) \tag{3.32}$$

where T is the SIFT token of the respective landmarks.

A landmark k is considered a match to landmark j if all of the following three conditions apply:

1. the correlation for landmark k is the smallest of all correlations computed for landmark j,
2. the correlation for landmark k is larger than the sift minimum correlation value, and
3. the correlation for landmark k is smaller than the second smallest correlation multiplied by the sift feature distinction value.

When landmark matching his complete the results are applied as measurements to the secondary local filter along with range measurements when available.

3.6 Range Measurements

In addition to landmark measurements, the secondary local filter may receive range measurements to other platforms cooperating in the navigation system. A measurement update requires three variables; the measurement, an error state to measurement state transformation matrix, and measurement covariance. The measurement is the direct output of the measurement device, while the transformation matrix and covariance are calculated from the position and covariance of the two platforms involved.

A measurement of the range between two platforms (z), regardless of the measurement device, is the root-sum-squared (RSS) difference in position ($\mathbf{p}$) of the two platforms plus a measurement error (v):

$$\begin{aligned} z &= |\Delta \mathbf{p}_{ab}| + v & (3.33) \\ &= |\mathbf{p}_a - \mathbf{p}_b| + v & (3.34) \end{aligned}$$

and may be expressed as an error measurement by:

$$\begin{aligned} \delta z &= \begin{bmatrix} \mathbf{H}_{aa} \ \mathbf{H}_{ab} \end{bmatrix} \begin{bmatrix} \delta \mathbf{p}_a \\ \delta \mathbf{p}_b \end{bmatrix} + v \\ &= \mathbf{H}_{aa} \delta \mathbf{p}_a + \mathbf{H}_{ab} \delta \mathbf{p}_b + v & (3.35) \end{aligned}$$

where a represents the platform performing the measurement, b represents the platform to which the measurement is made, $\mathbf{H}_{aa}$ is the transformation matrix from the state space of platform a to the measurement space of platform a, and $\mathbf{H}_{ab}$ is the transformation matrix from the state space of platform b to the measurement space of platform a.

The measurement and remote platforms each provide an estimate of their respective position. When these estimates are applied to equation (3.34),

$$\begin{aligned} \tilde{z} &= |\mathbf{p}_a^n - \mathbf{p}_b^n| + v & (3.36) \\ &= \sqrt{(x_a + x_b)^2 + (y_a + y_b)^2 + (z_a + z_b)^2} + v & (3.37) \end{aligned}$$

the range measurement is equal to the estimated range plus the measurement error. Therefore, the estimated range is:

$$\bar{z} = \sqrt{(x_a + x_b)^2 + (y_a + y_b)^2 + (z_a + z_b)^2} \tag{3.38}$$

The error to measurement state transformation matrices are calculated from the partial derivatives of the range estimate with respect to the position coordinates of each platform.

$$\mathbf{H}_{aa} = \left[\frac{\partial \bar{z}}{\partial x_a} \; \frac{\partial \bar{z}}{\partial y_a} \; \frac{\partial \bar{z}}{\partial z_a} \right] \Bigg|_{x=\tilde{x}} \tag{3.39}$$

$$\mathbf{H}_{ab} = \left[\frac{\partial \bar{z}}{\partial x_b} \; \frac{\partial \bar{z}}{\partial y_b} \; \frac{\partial \bar{z}}{\partial z_b} \right] \Bigg|_{x=\tilde{x}} \tag{3.40}$$

The partial derivative of the range estimate with respect to the x coordinate of platform a is the difference between the two platform x coordinates divided by the range estimate;

$$\frac{\partial \bar{z}}{\partial x_a} \Bigg|_{x=\tilde{x}} = \frac{x_a - x_b}{\sqrt{(x_a + x_b)^2 + (y_a + y_b)^2 + (z_a + z_b)^2}} = \frac{x_a - x_b}{\bar{z}} \tag{3.41}$$

and the calculations for the remaining partial derivatives for platform a are similar;

$$\frac{\partial \bar{z}}{\partial y_a} \Bigg|_{x=\tilde{x}} = \frac{y_a - y_b}{\bar{z}} \tag{3.42a}$$

$$\frac{\partial \bar{z}}{\partial z_a} \Bigg|_{x=\tilde{x}} = \frac{z_a - z_b}{\bar{z}} \tag{3.42b}$$

while the partial derivatives for platform b are the negative of the corresponding partial derivatives for platform a;

$$\left.\frac{\partial \bar{z}}{\partial x_b}\right|_{x=\tilde{x}} = \frac{x_b - x_a}{\bar{z}} \tag{3.42c}$$

$$\left.\frac{\partial \bar{z}}{\partial y_b}\right|_{x=\tilde{x}} = \frac{y_b - y_a}{\bar{z}} \tag{3.42d}$$

$$\left.\frac{\partial \bar{z}}{\partial z_b}\right|_{x=\tilde{x}} = \frac{z_b - z_a}{\bar{z}} \tag{3.42e}$$

The final variable needed for a range measurement is the measurement covariance. As the state estimates the position error of platform a, the error of the measurement device is redefined as:

$$\mathbf{v}^* = \mathbf{H}_{ab}\delta\mathbf{p}_b + v \tag{3.43}$$

and the error measurement becomes:

$$\delta\tilde{\mathbf{z}} = \mathbf{H}_{aa}\delta\mathbf{p}_a + \mathbf{v}^* \tag{3.44}$$

The error measurement covariance is computed as the expected value of the measurement device error:

$$\mathbf{R}^* = E\left[\mathbf{v}^*\mathbf{v}^{*^T}\right] = E\left[\mathbf{H}_{ab}\delta\mathbf{p}_b\mathbf{p}_b^T\mathbf{H}_{ab}^T + \mathbf{H}_{ab}\mathbf{p}_b v + v^T\mathbf{p}_b^T\mathbf{H}_{ab}^T + vv^T\right] \tag{3.45}$$

$$= \mathbf{H}_{ab}E\left[\mathbf{p}_b\mathbf{p}_b^T\right]\mathbf{H}_{ab}^T + 0 + 0 + E[vv^T] \tag{3.46}$$

$$= \mathbf{H}_{ab}\mathbf{R}_{p_b p_b}\mathbf{H}_{ab}^T + R \tag{3.47}$$

The next section develops the secondary local filter algorithm using the matched landmark and possible range measurements.

3.7 Secondary Local Filter

The secondary local filter provides a platform with the ability to incorporate information about landmarks viewed by remote platforms and the relative positioning of platforms cooperating in the navigation system. The filter calculates the error state to measurement state transformation matrix, measurement covariance, and residuals then performs a measurement update.

The filter is simply the measurement update step of an EKF and operates as described in Section 2.5 with one addition. Whenever a new landmark match is added to the measurement state of the filter, the process covariance must be reset to an initial condition. This procedure is performed by multiplying the process covariance matrix by the mask matrix, provided by the primary filter, and then adding the initialization matrix:

$$\mathbf{P}_i = \mathbf{P}_o\mathbf{M} + \mathbf{L} \tag{3.48}$$

where $\mathbf{P}_i$ is the covariance matrix after landmark initialization, $\mathbf{P}_o$ is the covariance matrix before landmark and initialization, $\mathbf{M}$ is the masked matrix, and $\mathbf{L}$ is the initialization matrix. The secondary local filter does not perform the propagation step of an EKF as the propagation is accomplished by the primary filter of the platform providing the measurement. Performing a second propagation will increase the error in the state estimate.

When the filter completes a measurement update the results are sent to the master filter for fusion with the primary filter results.

3.8 Master Filter

The master filter is designed to fuse the results of the primary and secondary filters. The implementation of the master filter is dependent on the method of implementation of the federated filter, as discussed in Section 3.1, and the federated filter reset mode. The filter scales the process covariances, as required by the implementation method; fuses the error states and process covariances; and resets all platform filters, as required by the reset mode.

3.8.1 Covariance Scaling. Scaling of the process covariances is performed only when the post-scaling method is implemented. The scaling is accomplished by multiplying the process covariances of the primary and secondary filters by the scaling factor (γ):

$$\mathbf{P}_p^s = \gamma\,\mathbf{P}_p \tag{3.49}$$

$$\mathbf{P}_s^s = \gamma\,\mathbf{P}_s \tag{3.50}$$

where the superscript s indicates the scaled covariance, subscript p indicates the primary filter, and subscript s indicates the secondary filter.

3.8.2 Information Fusion. Information fusion is performed using the information form of federated filter as this form is the simplest to implement and does not slow down the processing of the filter. Applying the additive information algorithm (2.85), presented in Section 2.7.1, provides:

$$\mathbf{P}^{-1} = \mathbf{P}_m^{-1} + \mathbf{P}_p^{-1} + \mathbf{P}_s^{-1} \tag{3.51}$$

$$\mathbf{P}^{-1}\hat{\mathbf{x}} = \mathbf{P}_m^{-1}\hat{\mathbf{x}}_m + \mathbf{P}_p^{-1}\hat{\mathbf{x}}_p + \mathbf{P}_s^{-1}\hat{\mathbf{x}}_s \tag{3.52}$$

when the filter is operated in fusion-reset mode, and:

$$\mathbf{P}^{-1} = \mathbf{P}_p^{-1} + \mathbf{P}_s^{-1} \tag{3.53}$$

$$\mathbf{P}^{-1}\hat{\mathbf{x}} = \mathbf{P}_p^{-1}\hat{\mathbf{x}}_p + \mathbf{P}_s^{-1}\hat{\mathbf{x}}_s \tag{3.54}$$

in no-reset mode. The subscripts m, p, and s represent the master, primary, and secondary filters respectively.

The covariance ($\mathbf{P}$) is calculated simply by inverting the information matrix ($\mathbf{P}^{-1}$). However, the error state vector ($\hat{\mathbf{x}}$) is found by pre-multiplying the results by the covariance;

$$\hat{\mathbf{x}} = \mathbf{P}\left(\mathbf{P}^{-1}\hat{\mathbf{x}}\right) \tag{3.55}$$

3.8.3 Filter Reset. The results of the information fusion are scaled and applied to the filters differently depending on the reset mode of operation.

For the no-reset mode the scaling parameter $\gamma = 1$ and the master filter is the only filter reset. In this mode the master filter only fuses the information provided by the primary and secondary filters and does not retain any information beyond the current time step. A failure or lack of measurements in the secondary filter causes the overall results to be identical to the original non-cooperative system.

The fusion-reset mode uses a scaling parameter $\gamma = 3$ and applies the results to all three filters. As a result each filter retains a portion of the information and the primary filter gains information improving the overall results if the secondary filter fails or does not receive measurements.

In the next chapter the multiple platform, cooperative navigation system developed here is validated using Monte Carlo simulation.

IV. Simulation Results and Observations

The cooperative, vision-aided navigation system, developed in Chapter III, is evaluated using `Matlab`® simulation and an incremental verification philosophy is applied to demonstrate the accuracy of the system. This chapter begins with a description of the filter setup, which is common to both implementation methods. The remainder of the chapter is sectioned into separate analyses of the post-scaling and pre-scaling methods presented in Section 3.1. Additionally, each section is divided into separate discussions of the simulation scenarios applied to the individual implementation methods.

All of the results presented in this chapter are for the first platform in the cooperative system. The results for additional platforms are presented in Appendices A-E.

4.1 Filter Setup

In this section the filter implementation common to all simulations is described.

4.1.1 Primary Filter. The primary local filter is implemented as an extended Kalman filter and estimates 15 platform states and 36 landmark states. The 15 platform states are composed of three states each representing the position, velocity, attitude, accelerometer biases, and a gyroscope biases. The 36 landmark states are composed of three position states for each of 12 landmarks tracked and estimated by the system.

Measurements are provided to the filter from a binocular vision system and are comprised of landmark positions within an image. The addition of new landmarks is limited to six landmarks per image. When the vision system is no longer able to find a tracked landmark, the corresponding states are propagated for one second before a new landmark is selected and tracked.

4.1.2 Secondary Filter. The secondary local filter is implemented as the measurement update step of a Kalman filter and estimates the same 51 platform and landmark states as the primary local filter. The federated filter allows the secondary filter to estimate additional landmark states, corresponding to extra landmark measurements from a second platform. However, this ability is not implemented as it requires adaptive scaling of the local filter information matrices and the master state vector and information matrix.

Landmark measurements are provided to the filter from other platforms and are limited to the 12 landmarks tracked by the primary filter.

Range measurements, between platforms, are generated as the actual distance between the platforms with an added random Gaussian error with:

$$\mu = 0\,m \tag{4.1}$$

$$\sigma = 5\,cm \tag{4.2}$$

The generated range measurements are not representative of any specific ranging sensor, and are adjusted to fit the scale used in the simulation scenarios.

4.1.3 Master Filter. The master filter estimates all of the states estimated by either of the local filters. As the secondary local filter is estimating the same states as the primary, the master filter state vector is composed of the same 51 states.

The local filter covariances are inverted to information matrices prior to fusion. This allows for simpler implementation with the greatly reduced possibility of implementation or coding errors.

4.2 Method 1: Post-Scaling

The post-scaling method, developed in Section 3.1, is simulated as a simple case comprised of two stationary platforms. Both platforms are placed along the east-west axis, $1\,m$ west and $1\,m$ east of the *n-frame* origin respectively. Thirty features, normally distributed in each direction, are centered about the coordinate $[10, 0, 0]^T$

with a standard deviation of $3\,m$. Figure 4.1 depicts the simulation scenario. The 50 run Monte Carlo simulation is performed over $200\,s$ at a $2\,Hz$ sampling rate.

The position, velocity and attitude errors relating to the *n-frame* north, east, and down directions are shown in Figures 4.2-4.10. As expected, the vision-aiding system constrains the drift of the INS. However, comparison of the cooperative systems to the non-cooperative system is difficult using separate plots of the errors. Therefore, the root-mean-squared (RMS) position, velocity and attitude errors are computed to provide one-dimensional metrics for the comparison of the three systems. The RMS errors are shown in Figures 4.11-4.21.

Using the post-scaling method to implement the cooperative systems displays no improvement of the navigation solution over the non-cooperative system. This simulation computes the solution of the primary filter at every time step prior to applying the results to the secondary and master filters. Therefore, there is no mechanism for applying the information, provided by additional platforms, to the primary filter. Without the addition of external information the system reverts to a non-cooperative state. This results is caused by either the predetermination of the primary result at every time step or subtle coding errors in the simulation.

The pre-scaling simulation overcomes the limitations of the post-scaling simulation by computing a total solution at each time step prior to continuing to the next. The results of this method are presented in the next section.

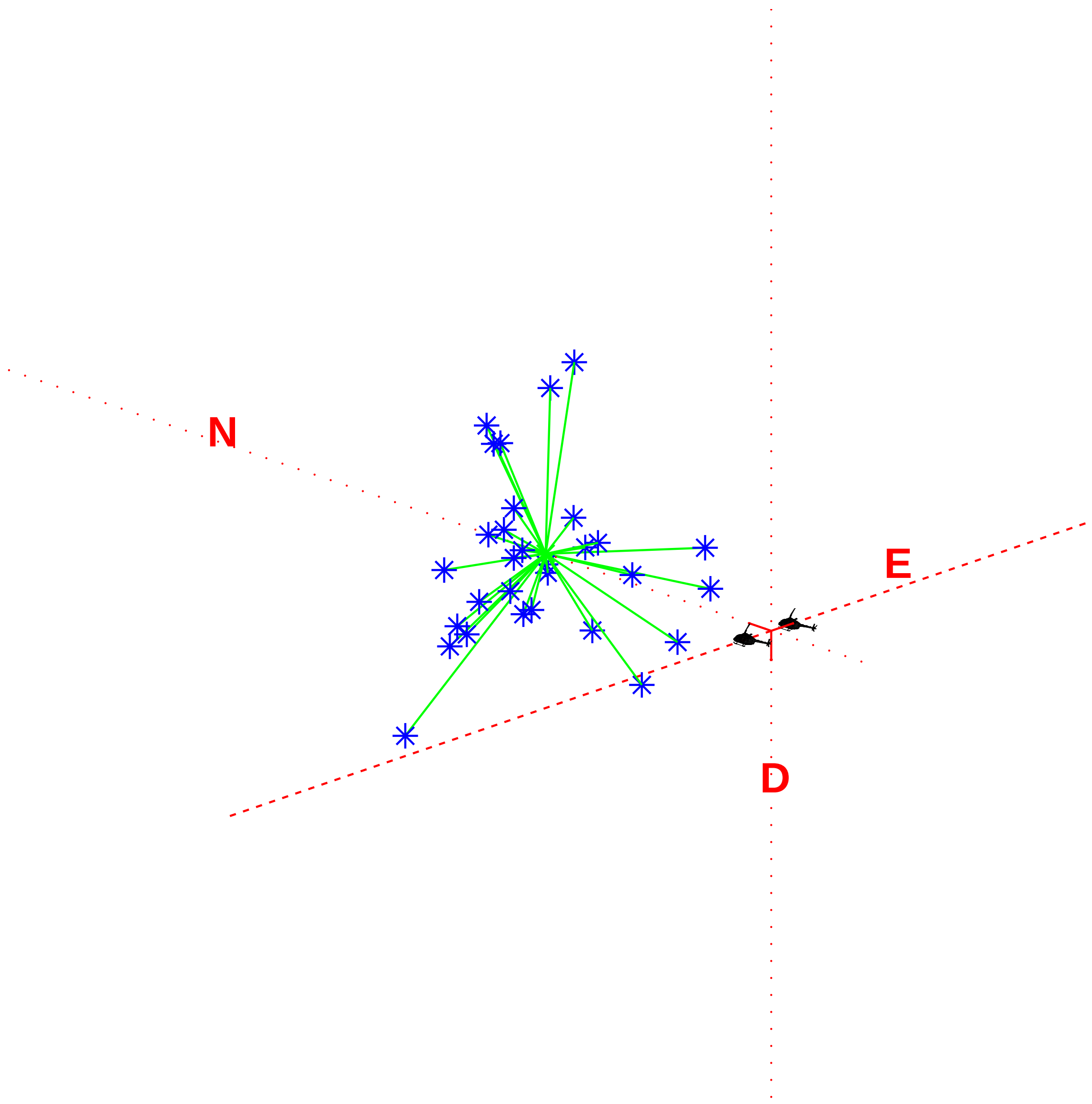
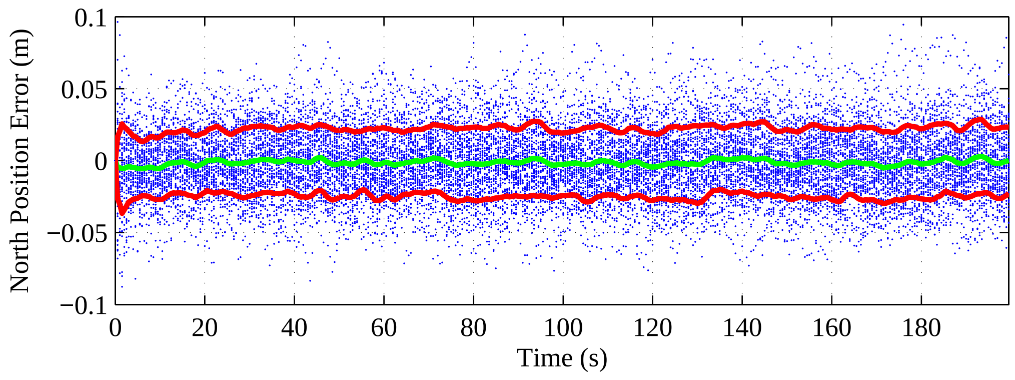
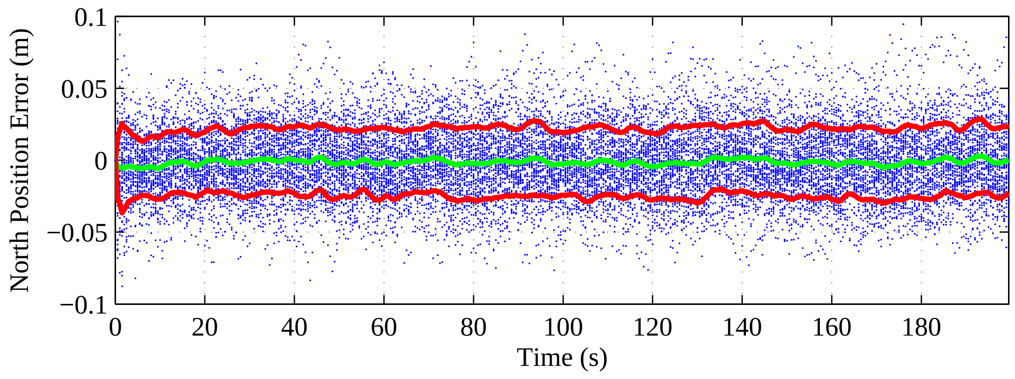


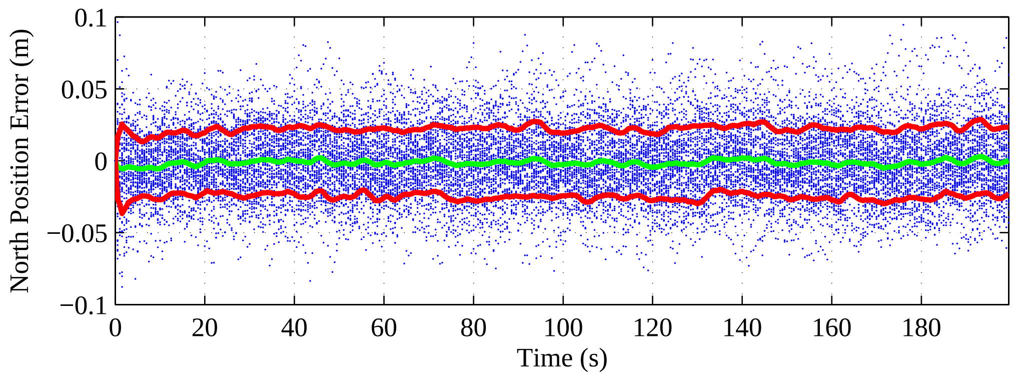
Figure 4.1: Two platform stationary simulation scenario using the post-scaling method. The platforms are each positioned $1\,m$ from the origin (red lines) along the east-west axis. Thirty random features (blue asterisk) are positioned around the coordinate $[10, 0, 0]^T$.



(a) Non-Cooperative Simulation

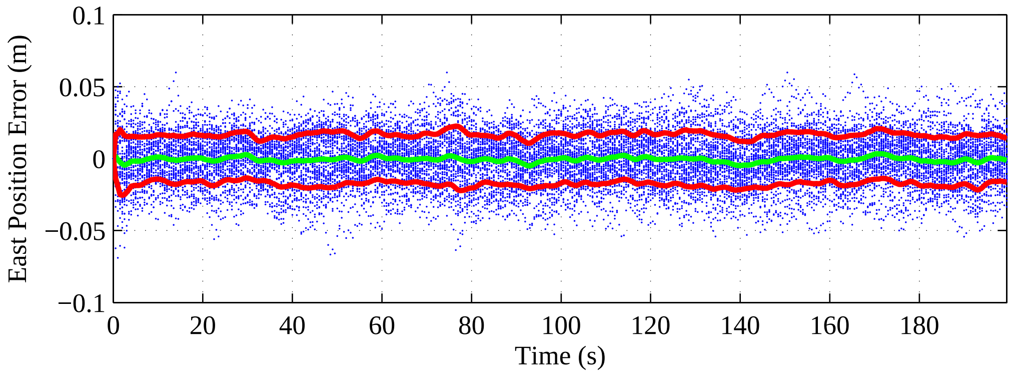


(b) Cooperative Simulation without Ranging

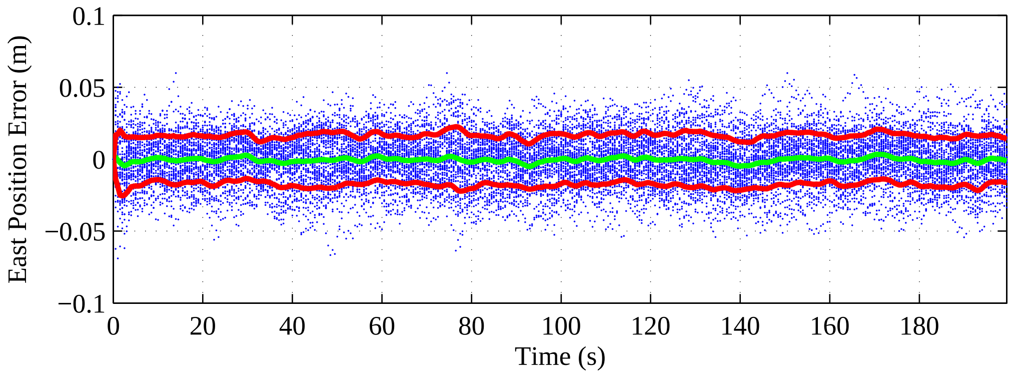


(c) Cooperative Simulation with Ranging

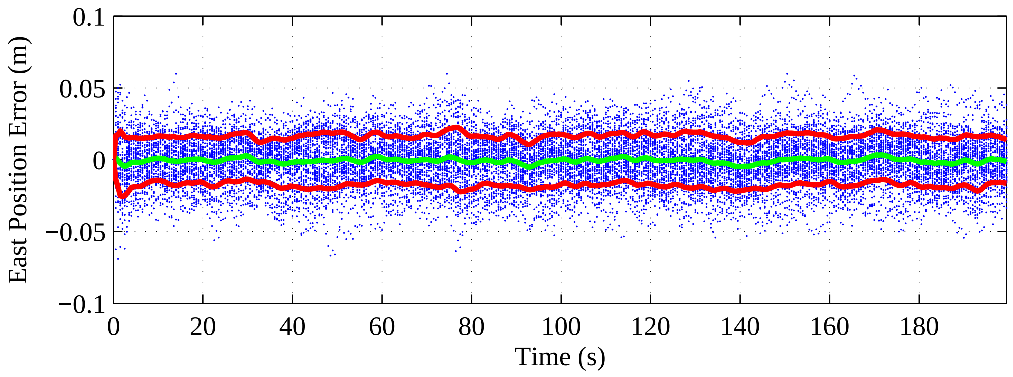
Figure 4.2: Simulated 50-run Monte Carlo position error results along the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (top), 2) cooperative without range measurements (middle), and 3) cooperative with range measurements (bottom). The position errors of each run are represented by the blue dotted lines. The ensemble mean is indicated by the green line while the ensemble standard deviation is indicated by the red lines.



(a) Non-Cooperative Simulation

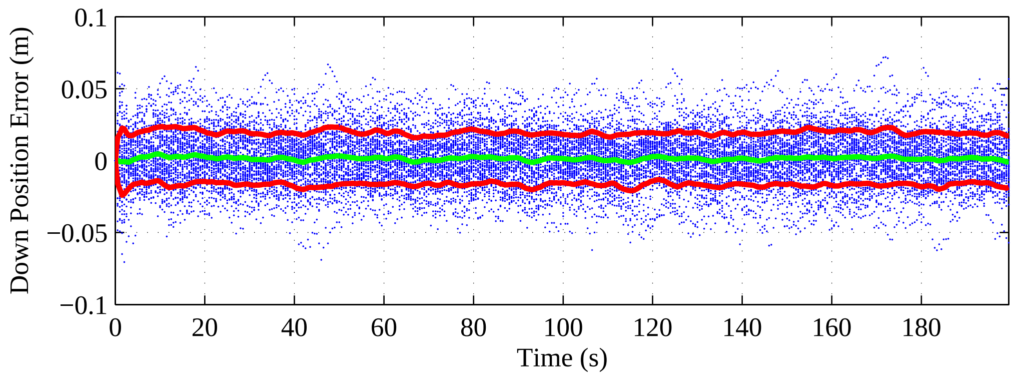


(b) Cooperative Simulation without Ranging

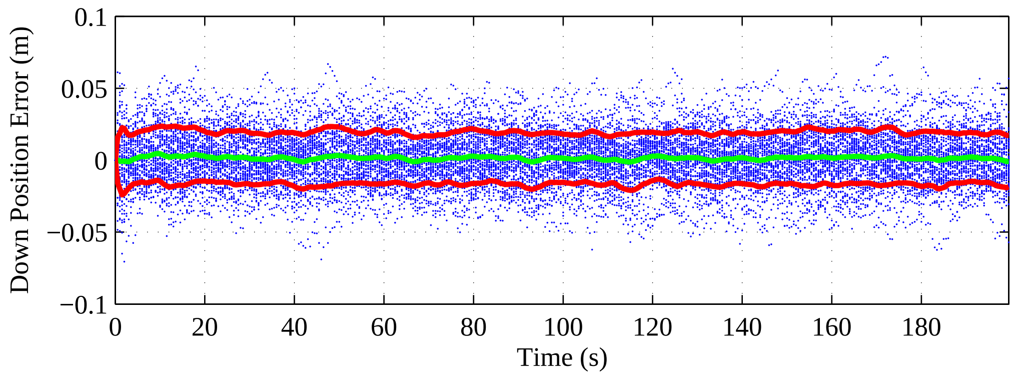


(c) Cooperative Simulation with Ranging

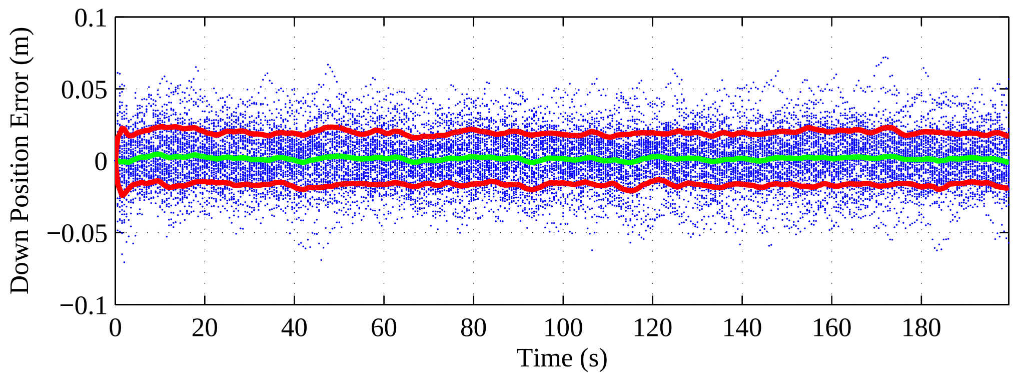
Figure 4.3: Simulated 50-run Monte Carlo position error results along the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (top), 2) cooperative without range measurements (middle), and 3) cooperative with range measurements (bottom). The position errors of each run are represented by the blue dotted lines. The ensemble mean is indicated by the green line while the ensemble standard deviation is indicated by the red lines.



(a) Non-Cooperative Simulation

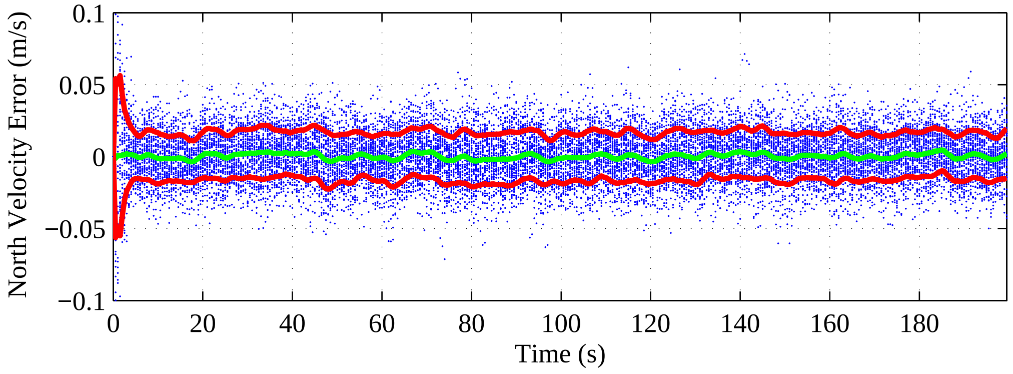


(b) Cooperative Simulation without Ranging

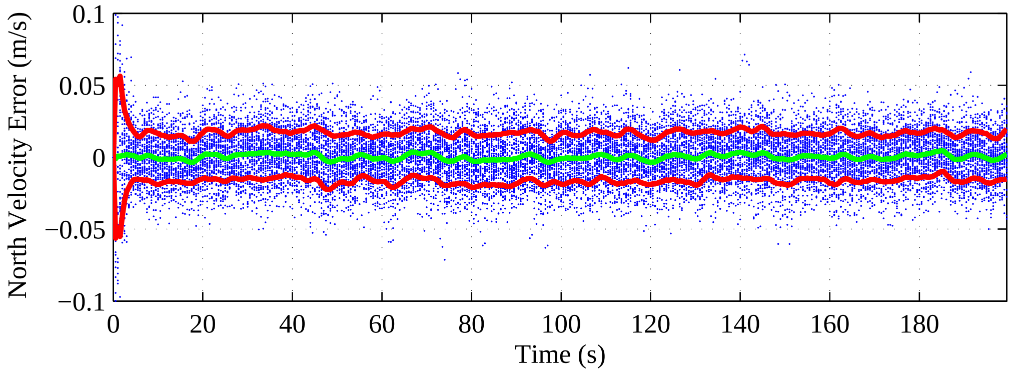


(c) Cooperative Simulation with Ranging

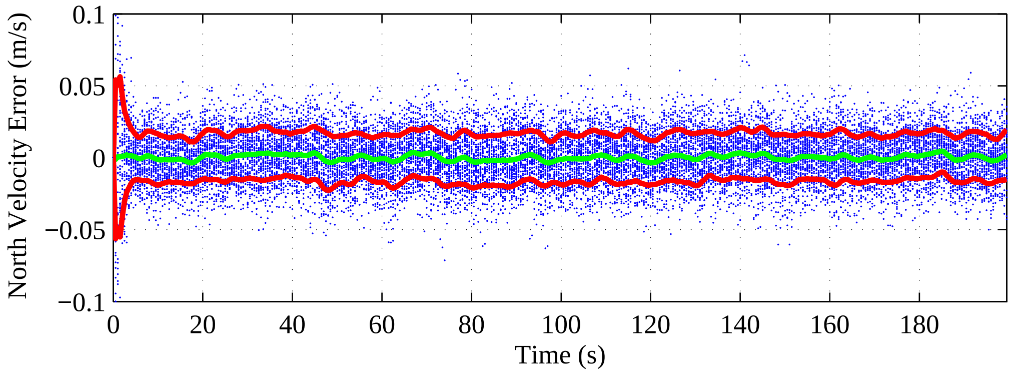
Figure 4.4: Simulated 50-run Monte Carlo position error results along the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (top), 2) cooperative without range measurements (middle), and 3) cooperative with range measurements (bottom). The position errors of each run are represented by the blue dotted lines. The ensemble mean is indicated by the green line while the ensemble standard deviation is indicated by the red lines.



(a) Non-Cooperative Simulation

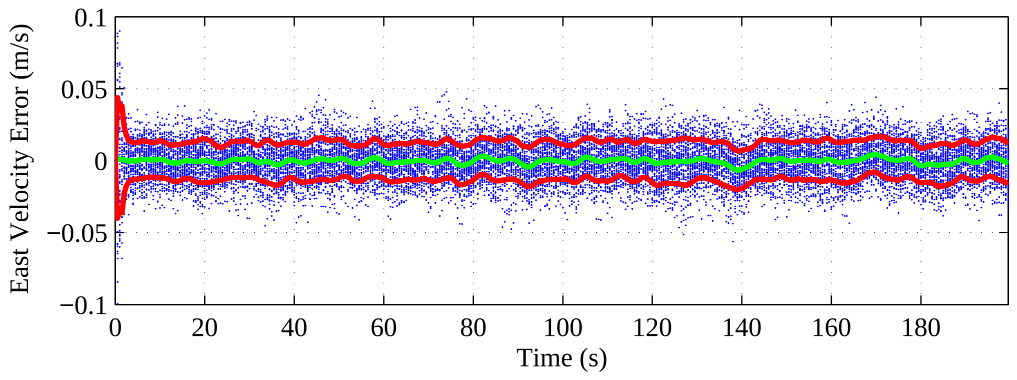


(b) Cooperative Simulation without Ranging

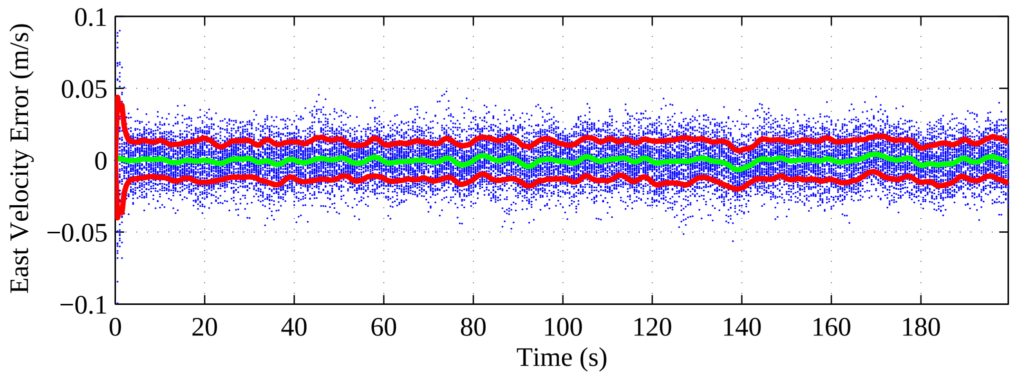


(c) Cooperative Simulation with Ranging

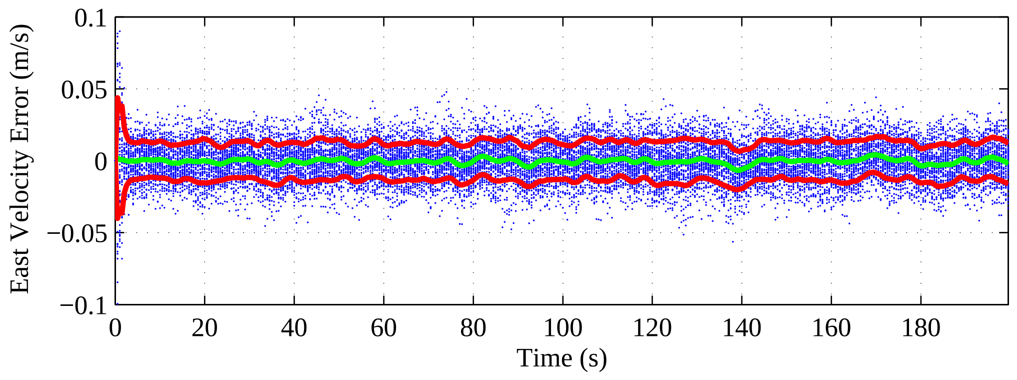
Figure 4.5: Simulated 50-run Monte Carlo velocity error results along the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (top), 2) cooperative without range measurements (middle), and 3) cooperative with range measurements (bottom). The velocity errors of each run are represented by the blue dotted lines. The ensemble mean is indicated by the green line while the ensemble standard deviation is indicated by the red lines.



(a) Non-Cooperative Simulation

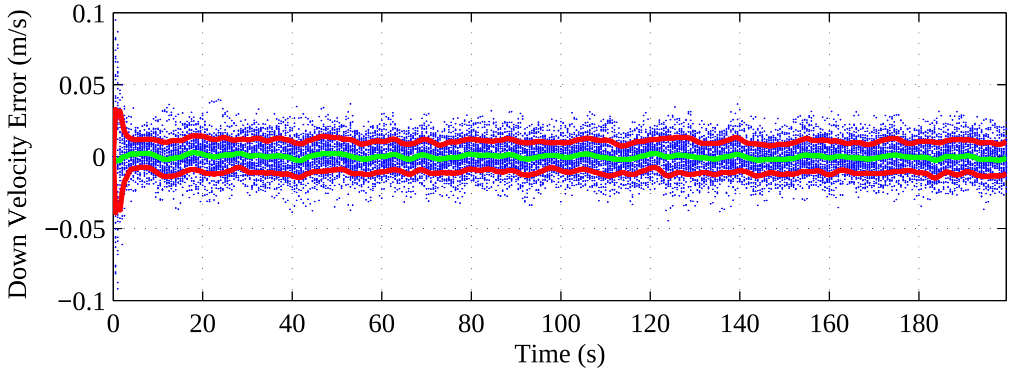


(b) Cooperative Simulation without Ranging

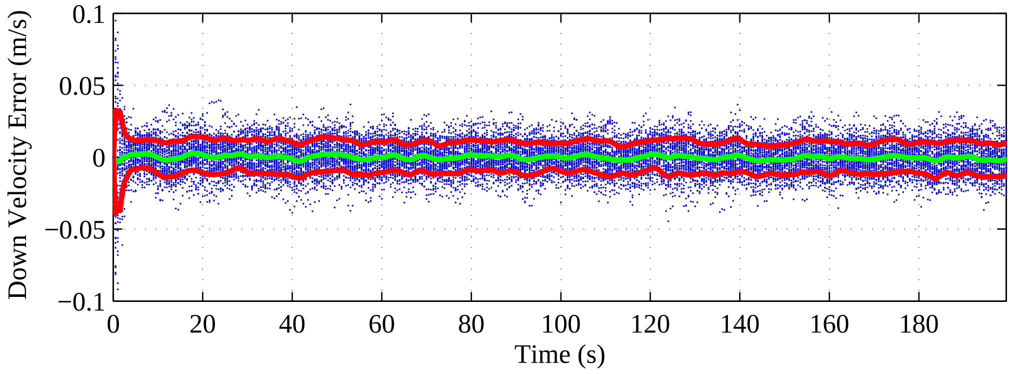


(c) Cooperative Simulation with Ranging

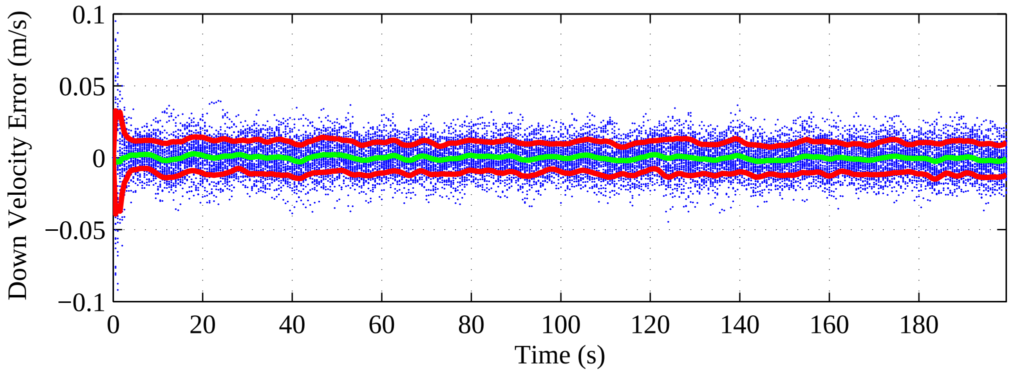
Figure 4.6: Simulated 50-run Monte Carlo velocity error results along the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (top), 2) cooperative without range measurements (middle), and 3) cooperative with range measurements (bottom). The velocity errors of each run are represented by the blue dotted lines. The ensemble mean is indicated by the green line while the ensemble standard deviation is indicated by the red lines.



(a) Non-Cooperative Simulation

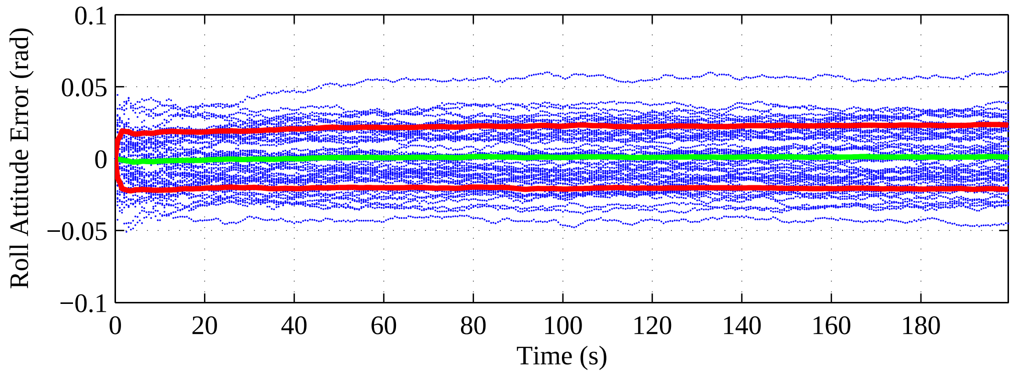


(b) Cooperative Simulation without Ranging

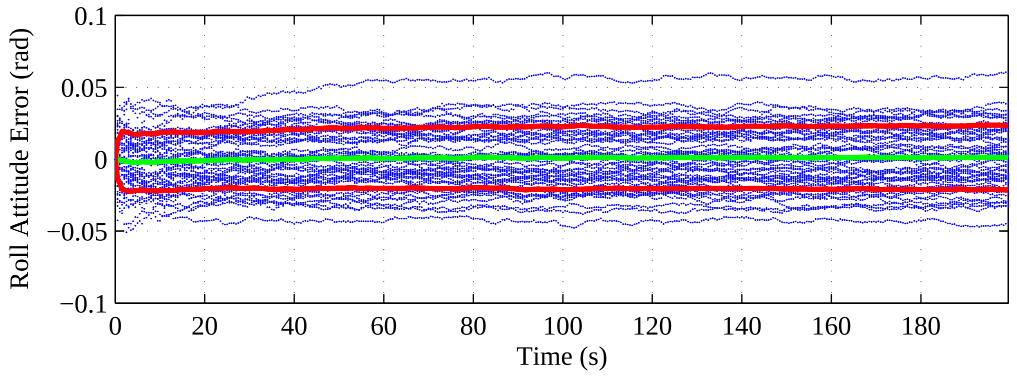


(c) Cooperative Simulation with Ranging

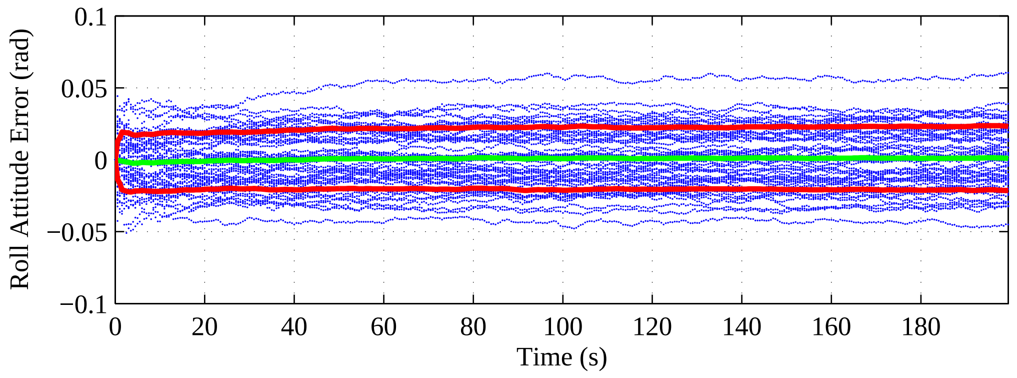
Figure 4.7: Simulated 50-run Monte Carlo velocity error results along the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (top), 2) cooperative without range measurements (middle), and 3) cooperative with range measurements (bottom). The velocity errors of each run are represented by the blue dotted lines. The ensemble mean is indicated by the green line while the ensemble standard deviation is indicated by the red lines.



(a) Non-Cooperative Simulation

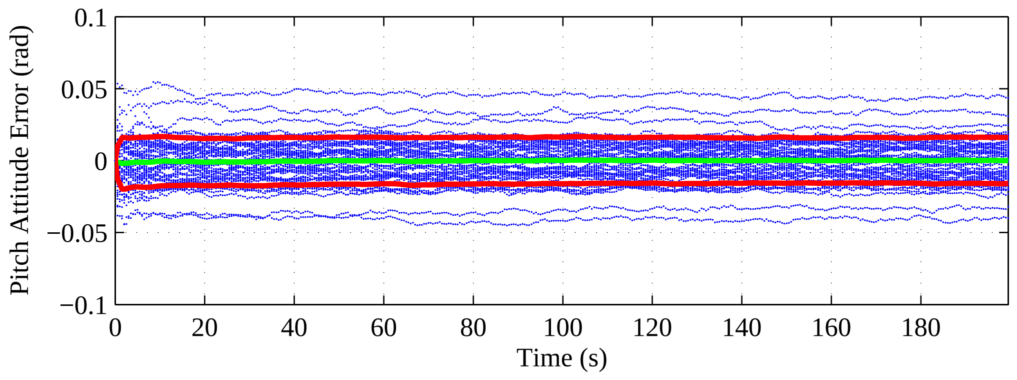


(b) Cooperative Simulation without Ranging

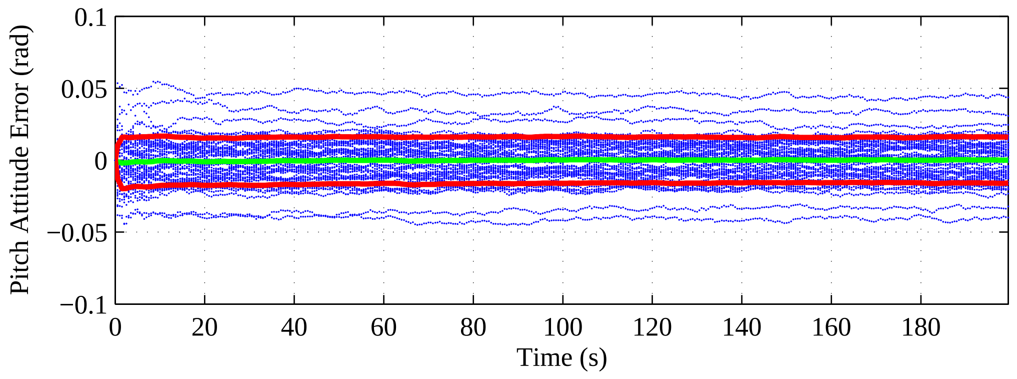


(c) Cooperative Simulation with Ranging

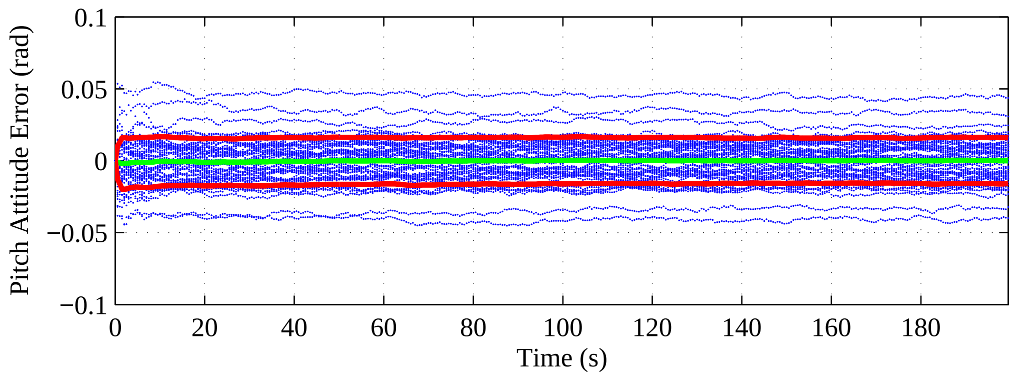
Figure 4.8: Simulated 50-run Monte Carlo attitude error results about the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (top), 2) cooperative without range measurements (middle), and 3) cooperative with range measurements (bottom). The attitude errors of each run are represented by the blue dotted lines. The ensemble mean is indicated by the green line while the ensemble standard deviation is indicated by the red lines.



(a) Non-Cooperative Simulation

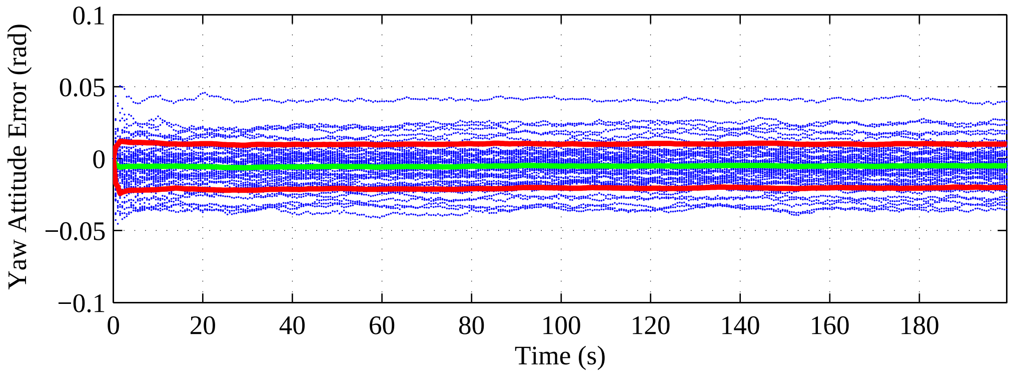


(b) Cooperative Simulation without Ranging

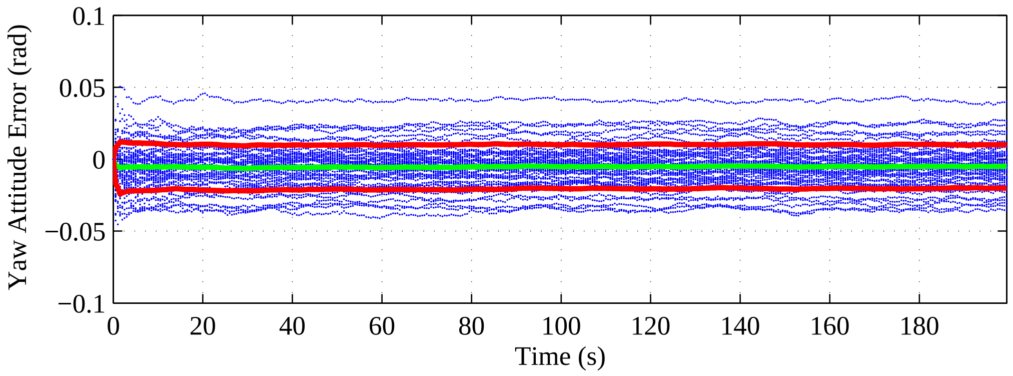


(c) Cooperative Simulation with Ranging

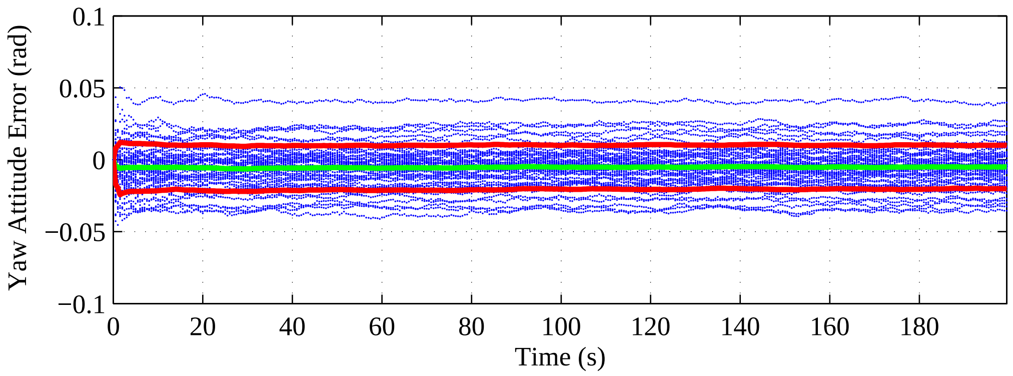
Figure 4.9: Simulated 50-run Monte Carlo attitude error results about the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (top), 2) cooperative without range measurements (middle), and 3) cooperative with range measurements (bottom). The attitude errors of each run are represented by the blue dotted lines. The ensemble mean is indicated by the green line while the ensemble standard deviation is indicated by the red lines.



(a) Non-Cooperative Simulation



(b) Cooperative Simulation without Ranging



(c) Cooperative Simulation with Ranging

Figure 4.10: Simulated 50-run Monte Carlo attitude error results about the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (top), 2) cooperative without range measurements (middle), and 3) cooperative with range measurements (bottom). The attitude errors of each run are represented by the blue dotted lines. The ensemble mean is indicated by the green line while the ensemble standard deviation is indicated by the red lines.

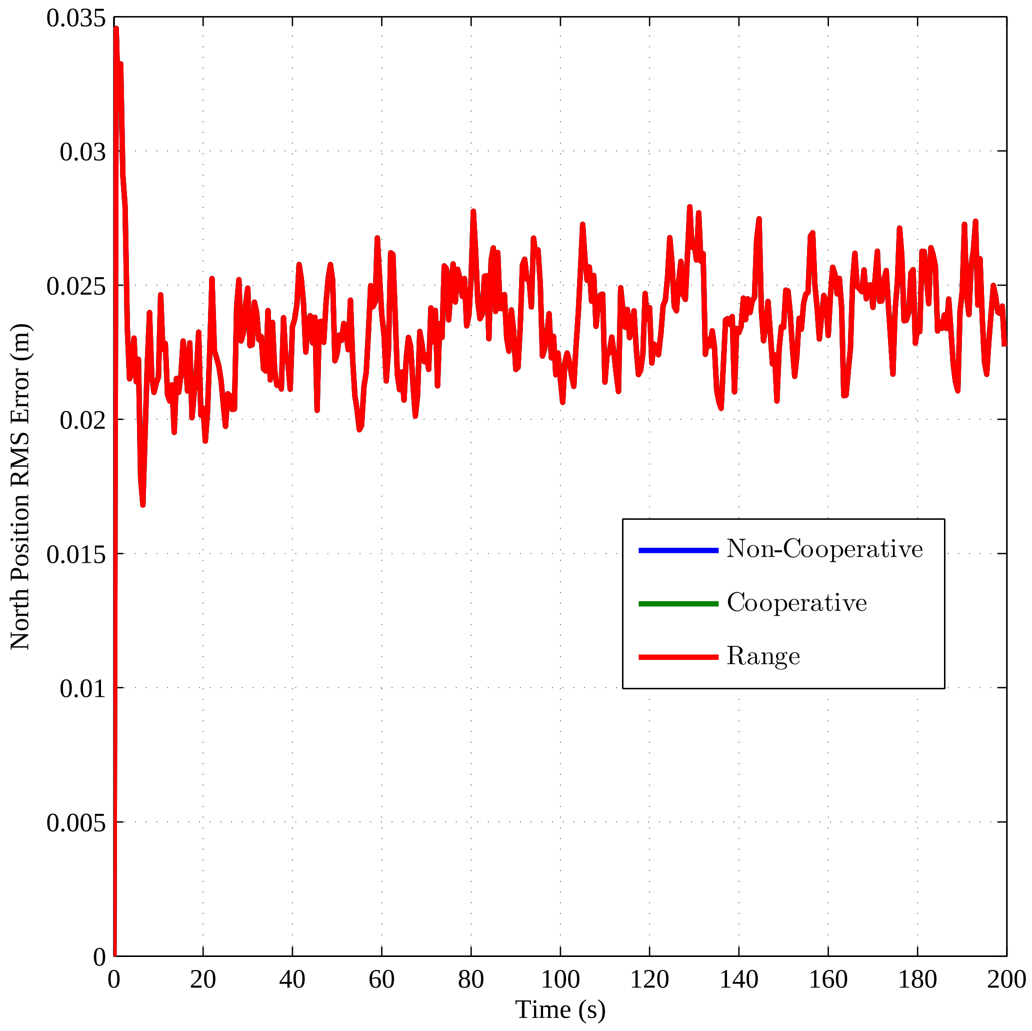


Figure 4.11: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

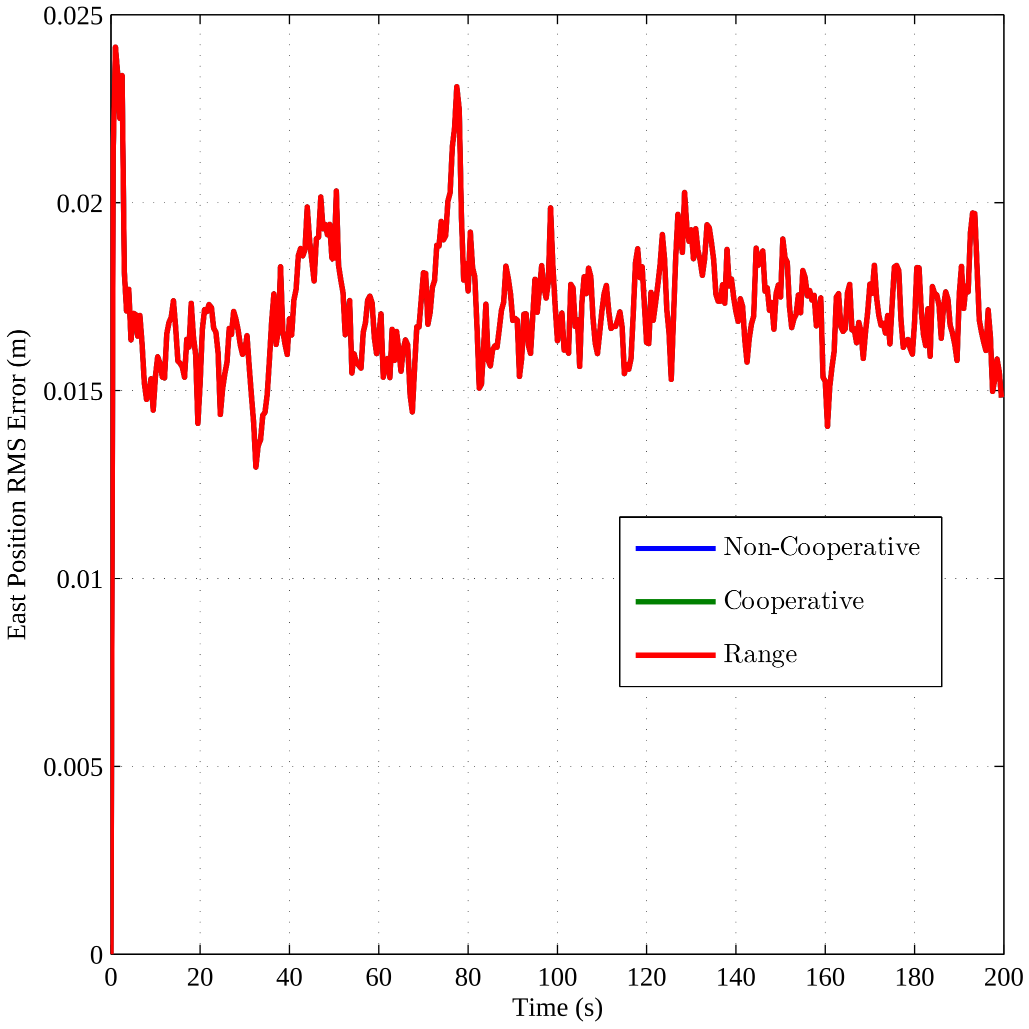


Figure 4.12: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

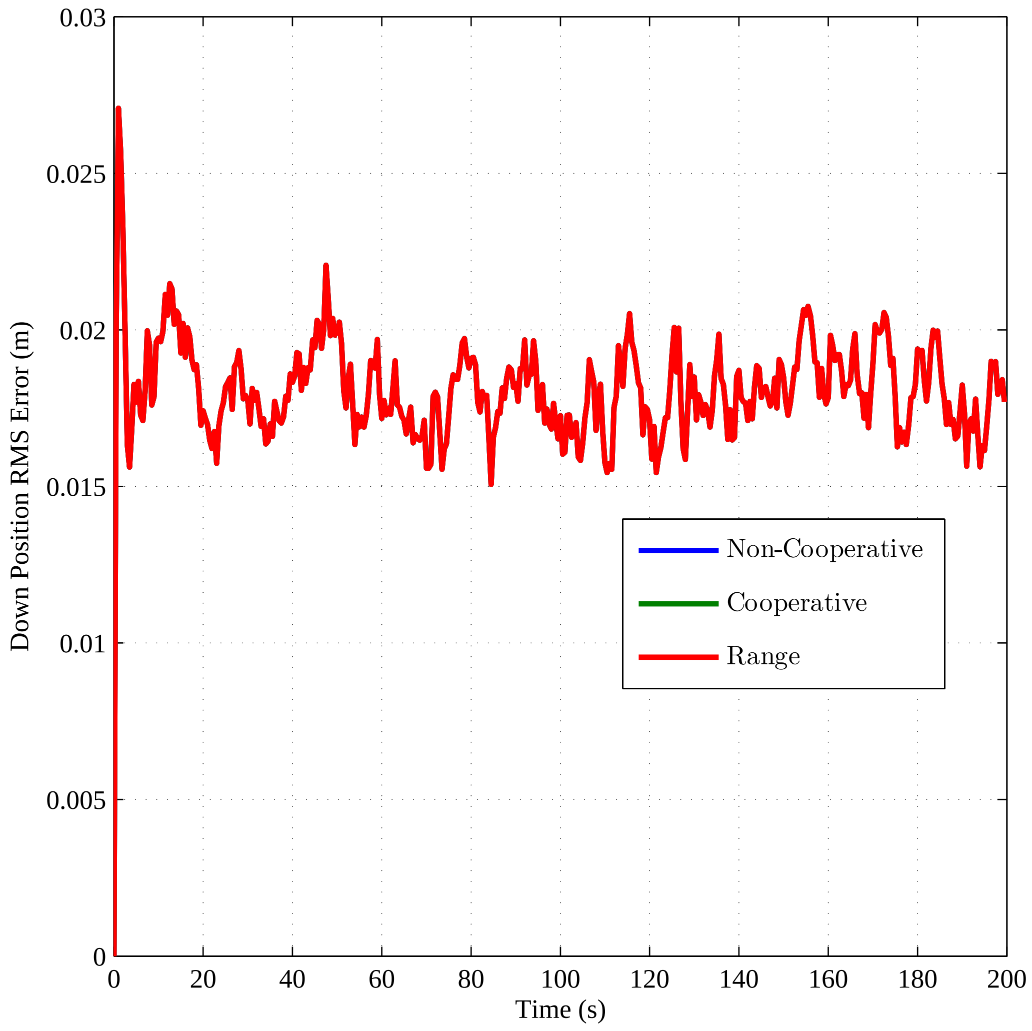


Figure 4.13: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

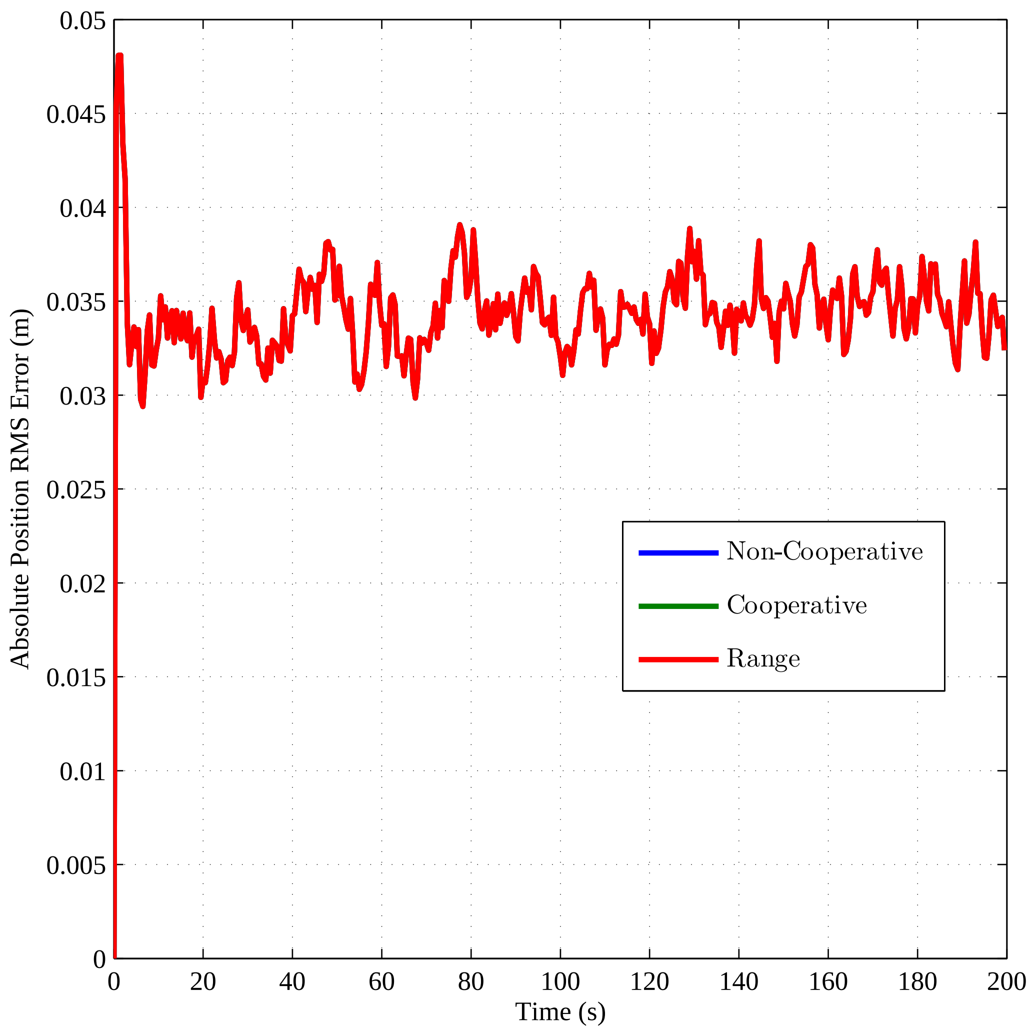


Figure 4.14: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute position error results. The results shown are for the first of two stationary platforms simulated using three system implementations: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

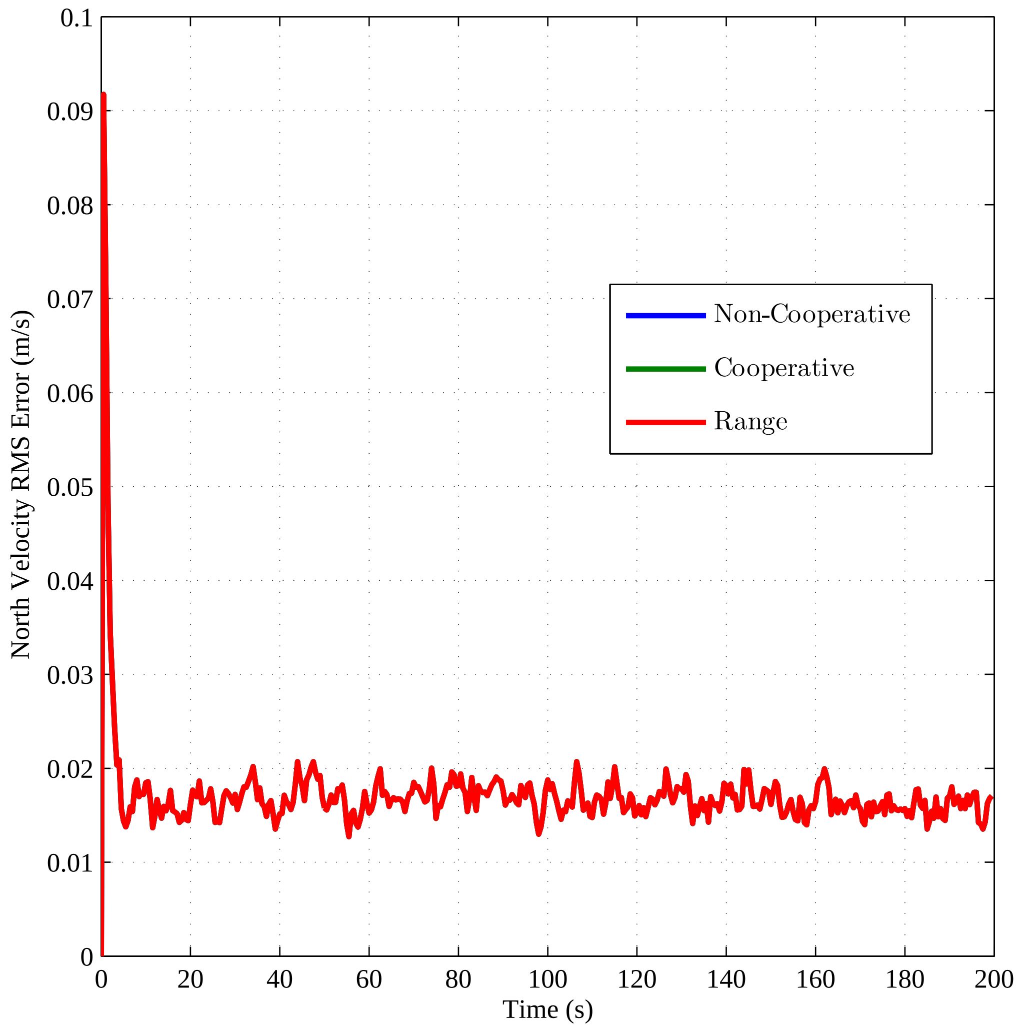


Figure 4.15: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

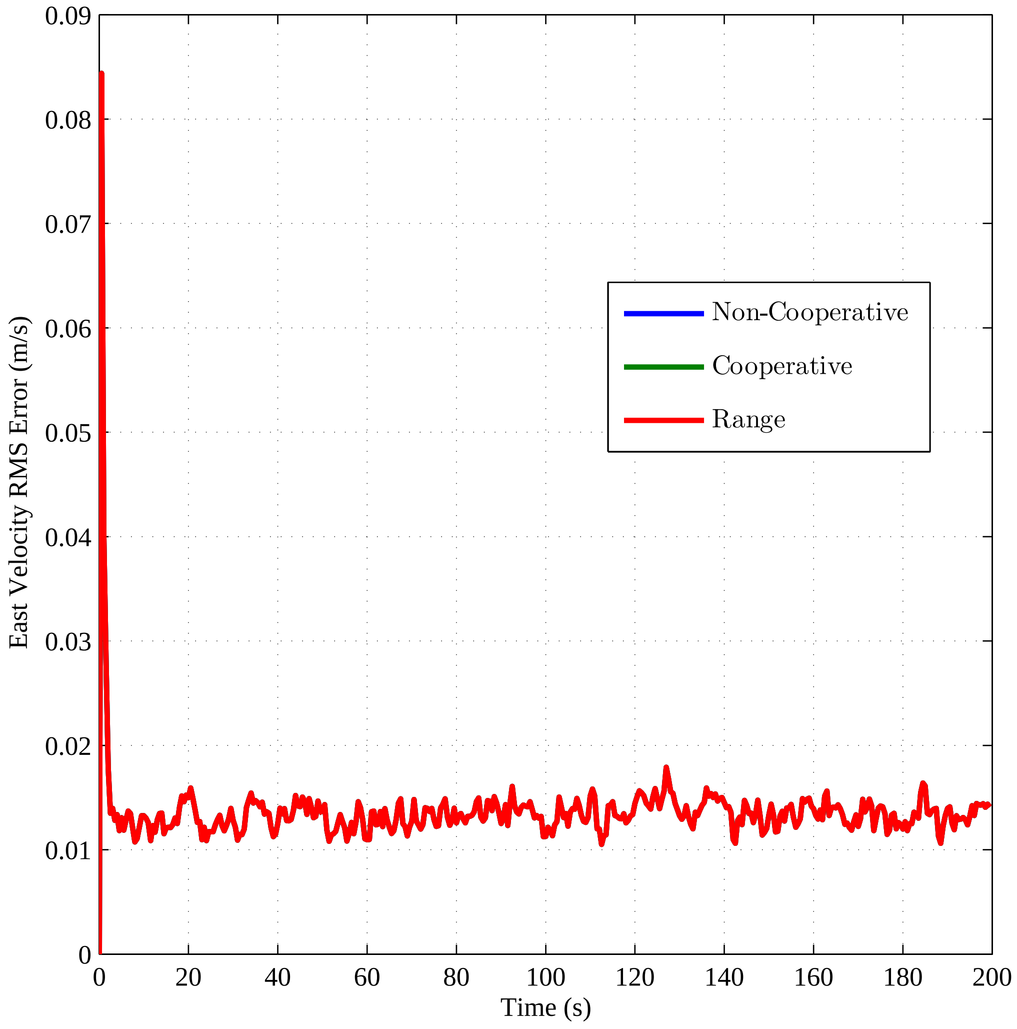


Figure 4.16: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

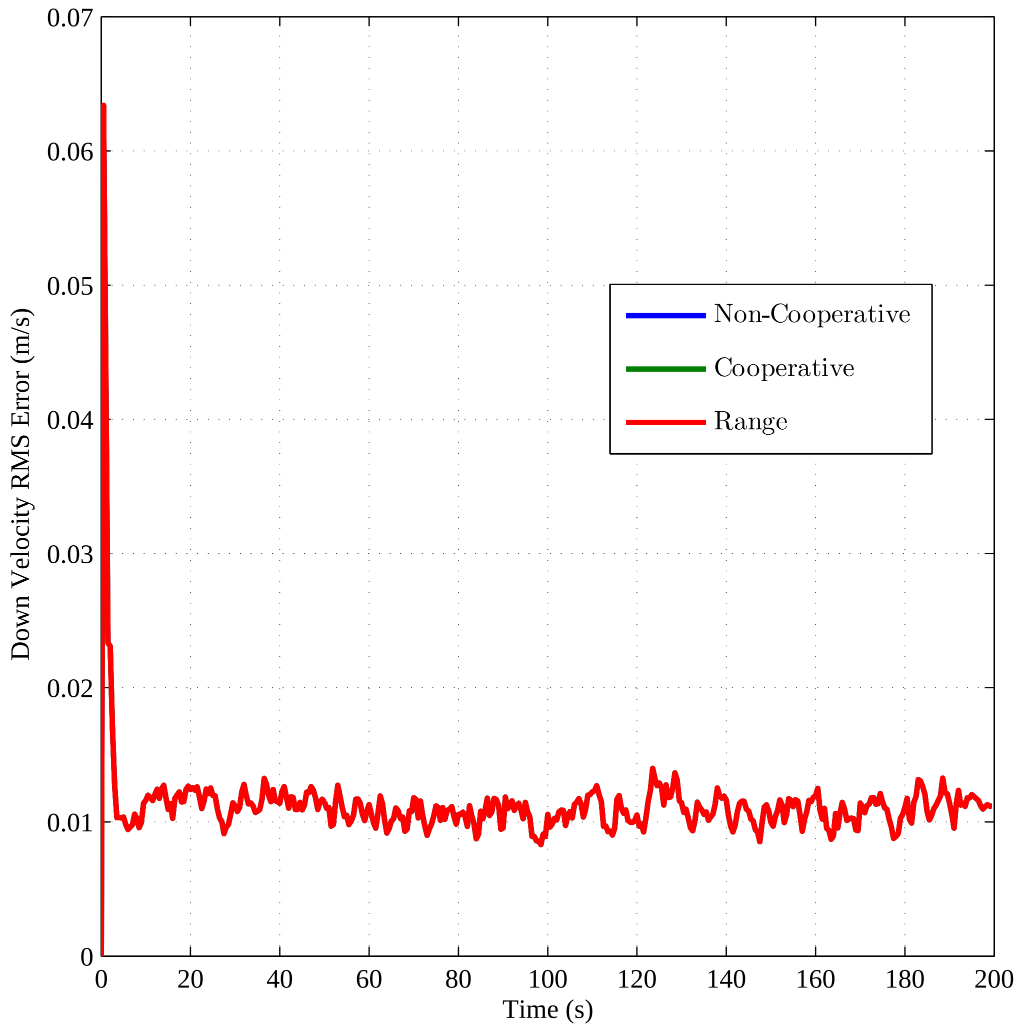


Figure 4.17: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

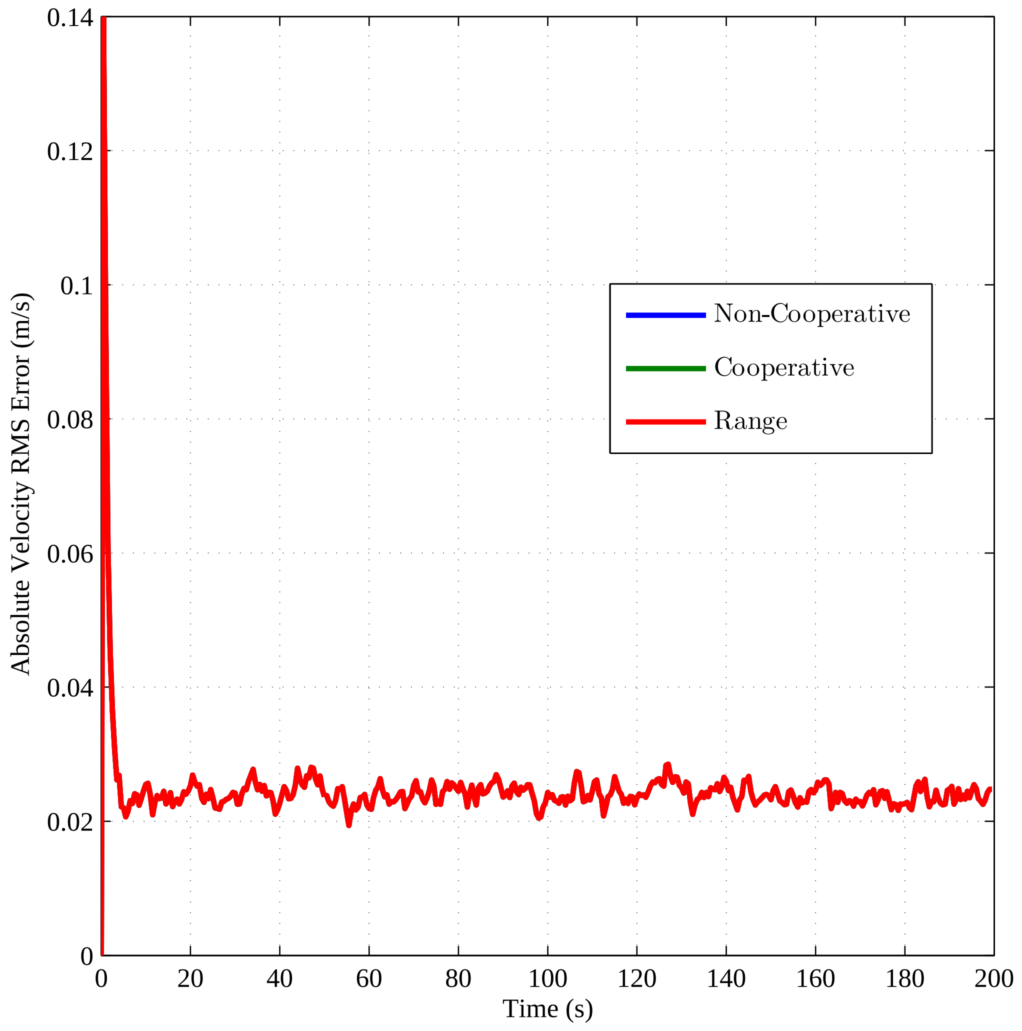


Figure 4.18: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute velocity error results . The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) co-operative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

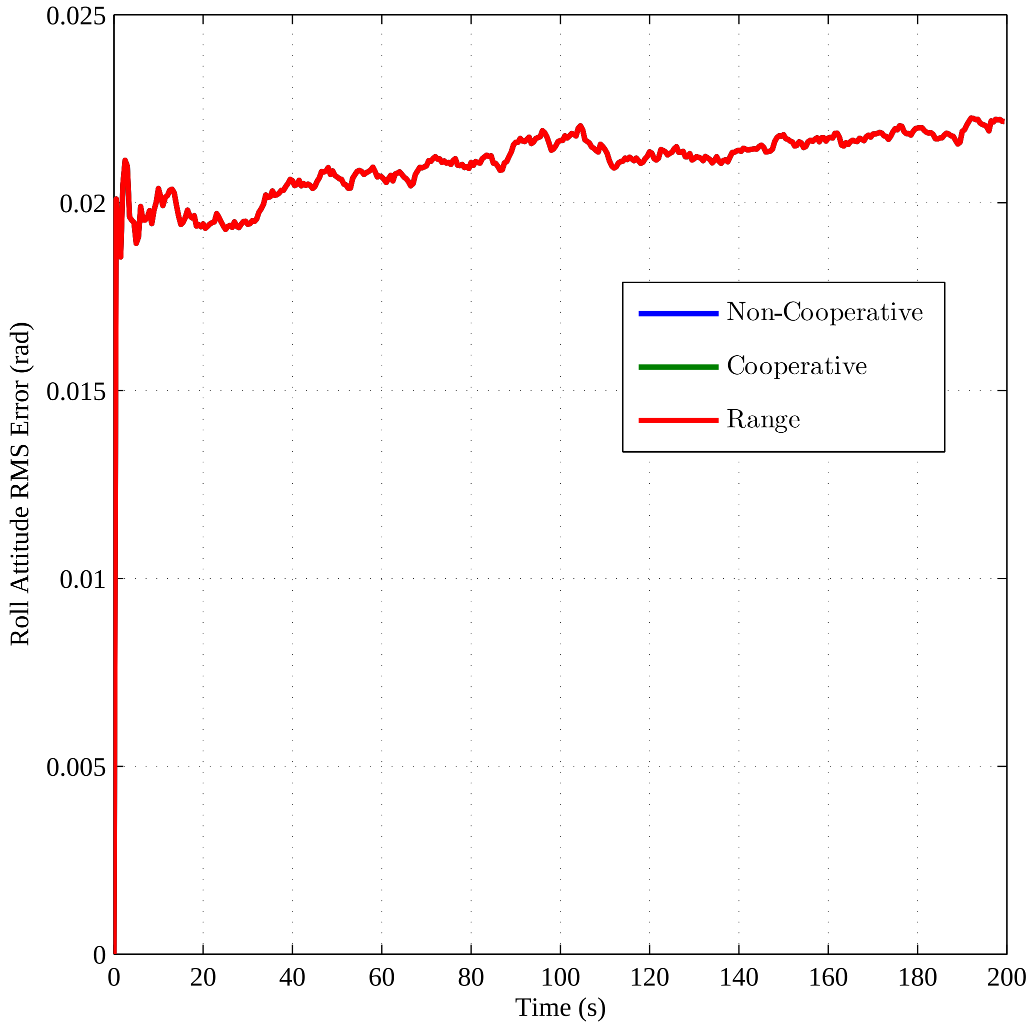


Figure 4.19: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

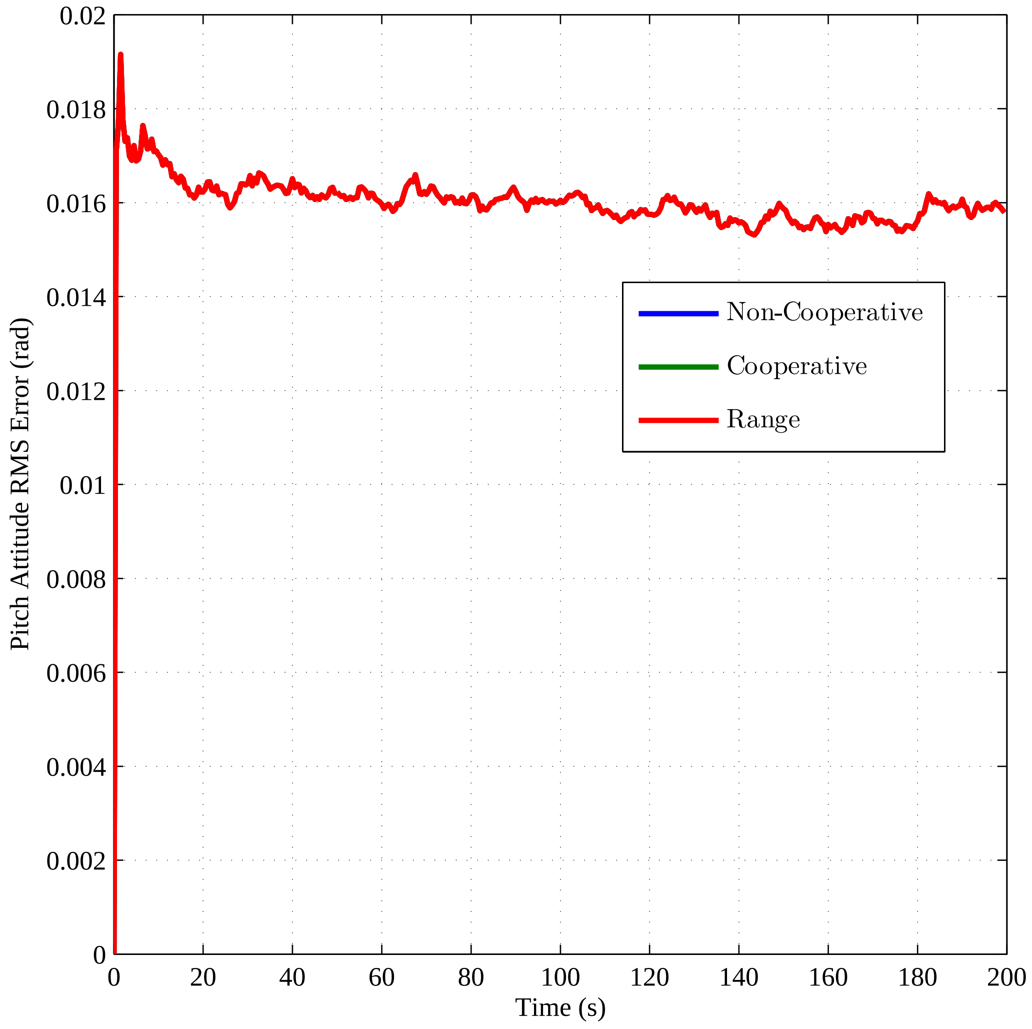


Figure 4.20: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

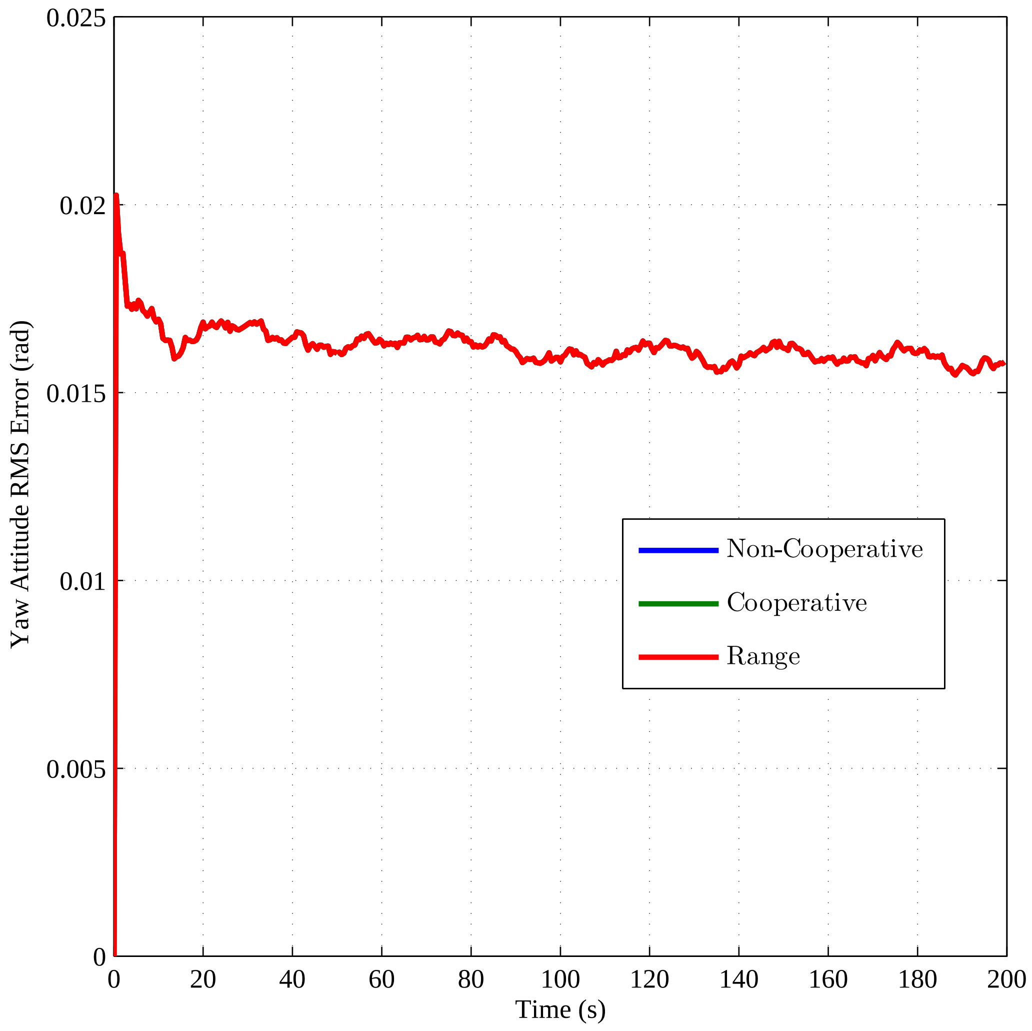


Figure 4.21: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

4.3 Method 2: Pre-Scaling

The pre-scaling algorithm, developed in Section 3.1, is implemented using four simulation scenarios: *1*) two stationary platforms, *2*) two platforms passing through a 25 m long hallway, *3*) two platforms orbiting a point in space, and *4*) two platforms performing a 360° yaw in a 10 m radius circular room.

4.3.1 Stationary Platforms.

The first simulation performed with the pre-scaling algorithm repeats the simulation performed using the post-scaling method. Figure 4.1 depicts the simulation scenario. The 50 run Monte Carlo simulation is performed over 200 s at a 2 Hz sampling rate.

The RMS errors for the stationary platform simulation are shown in Figures 4.22-4.32. As expected, the cooperative systems reduce the error of the navigation solution position and velocity. Additionally, the inclusion of ranging information reduces the error more than the standard cooperative system. Figure 4.25 provides a single indicator of the overall position performance of the filters. The cooperative systems reduce the absolute position error by approximately 0.5 cm without ranging information and 1 cm with ranging.

The cooperative systems do not improve the attitude error, as seen in Figures 4.30-4.32, and this additional error is due to the unchanging pose of the two platforms. The cooperative systems increase the attitude errors by as much as 0.01 radians, or 6°, in any one direction. The reasons for the added attitude error and a technique to correct it are discussed in Section 4.3.5.

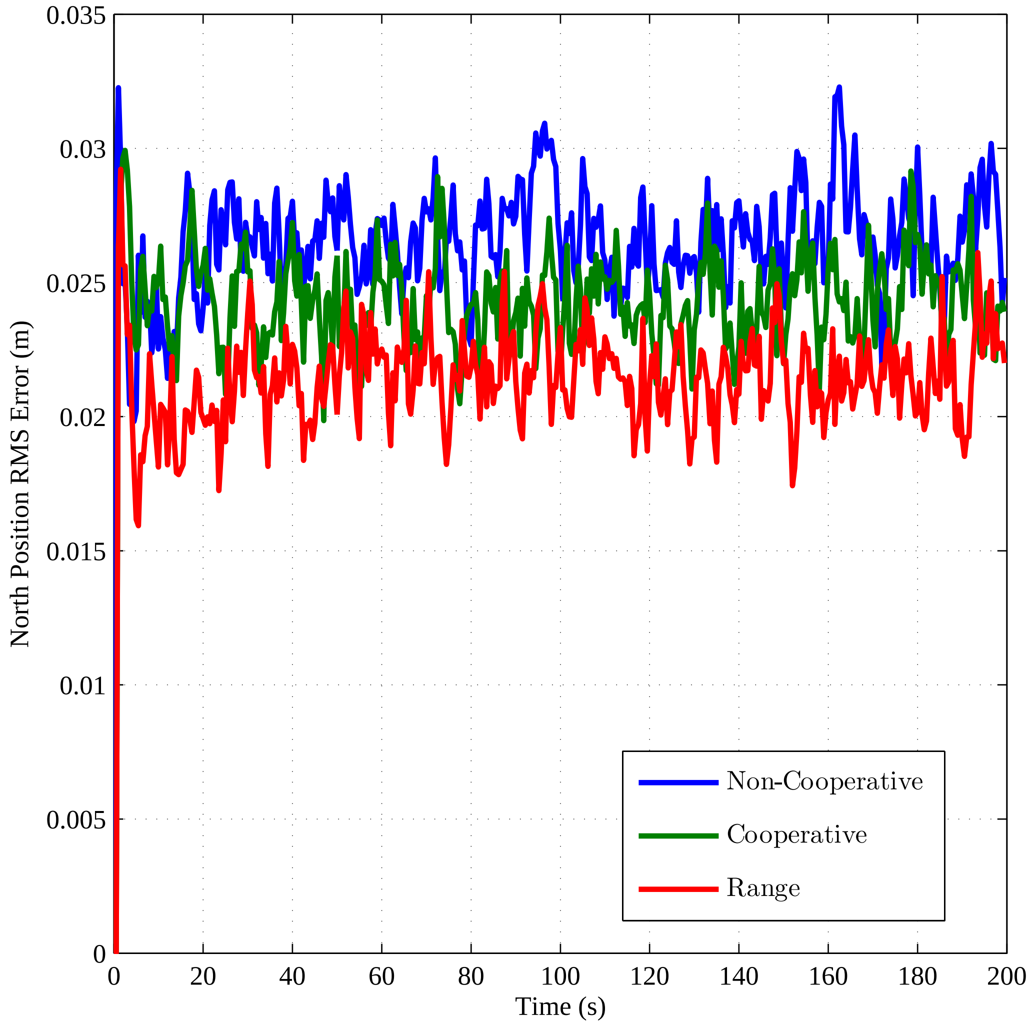


Figure 4.22: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

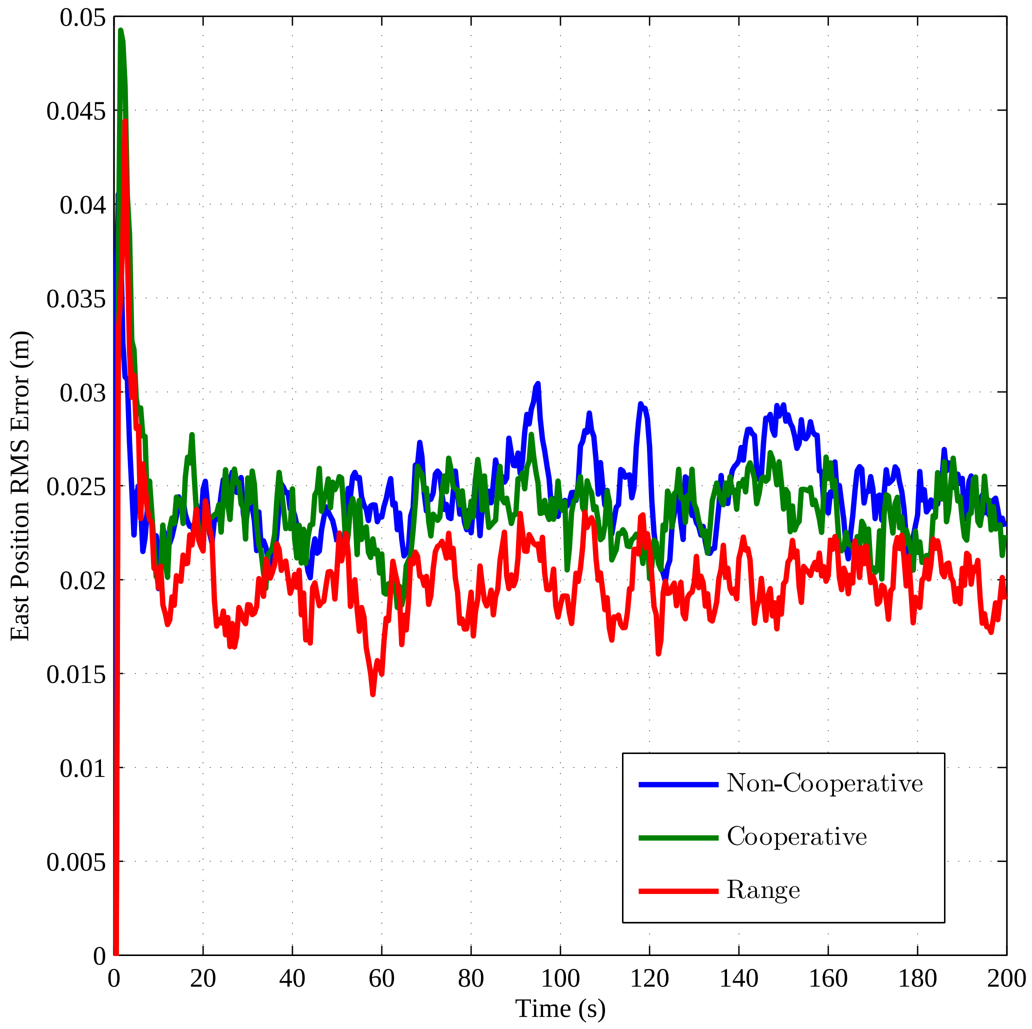


Figure 4.23: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

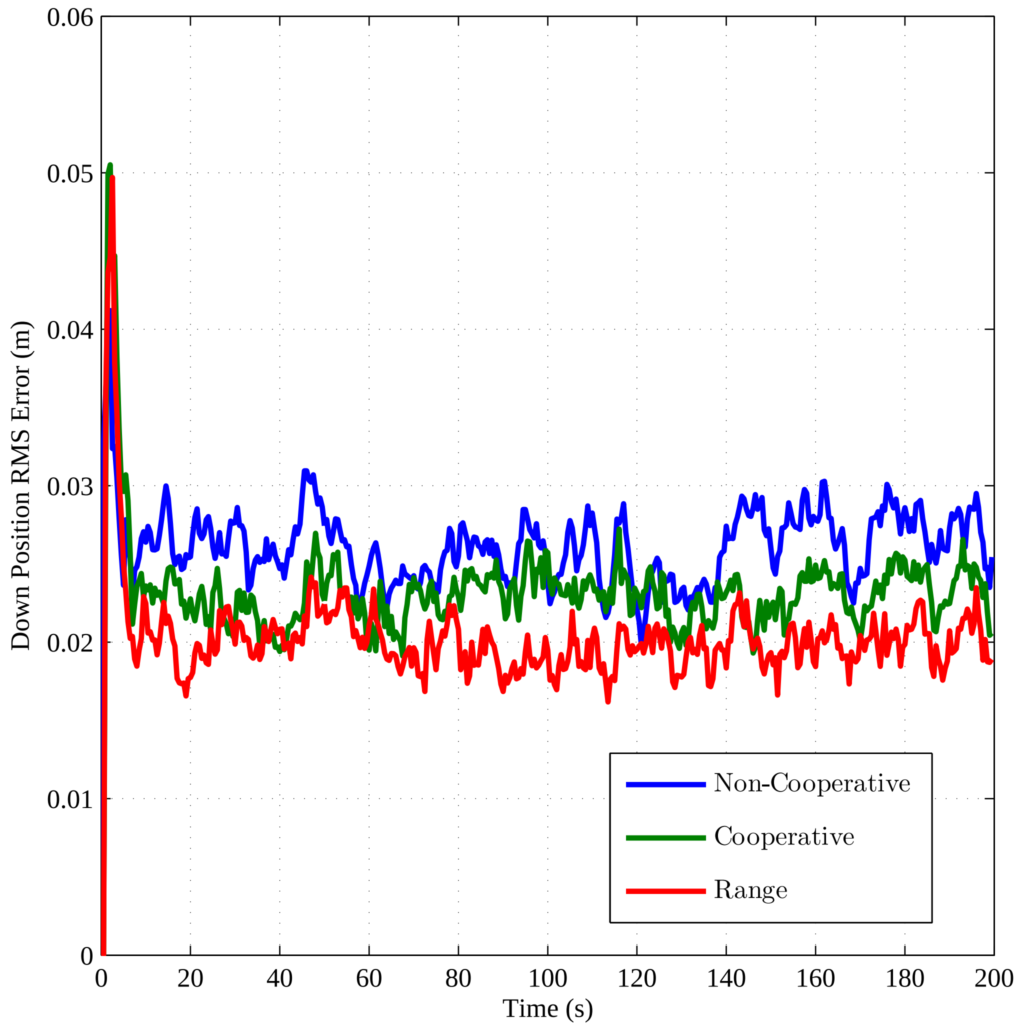


Figure 4.24: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

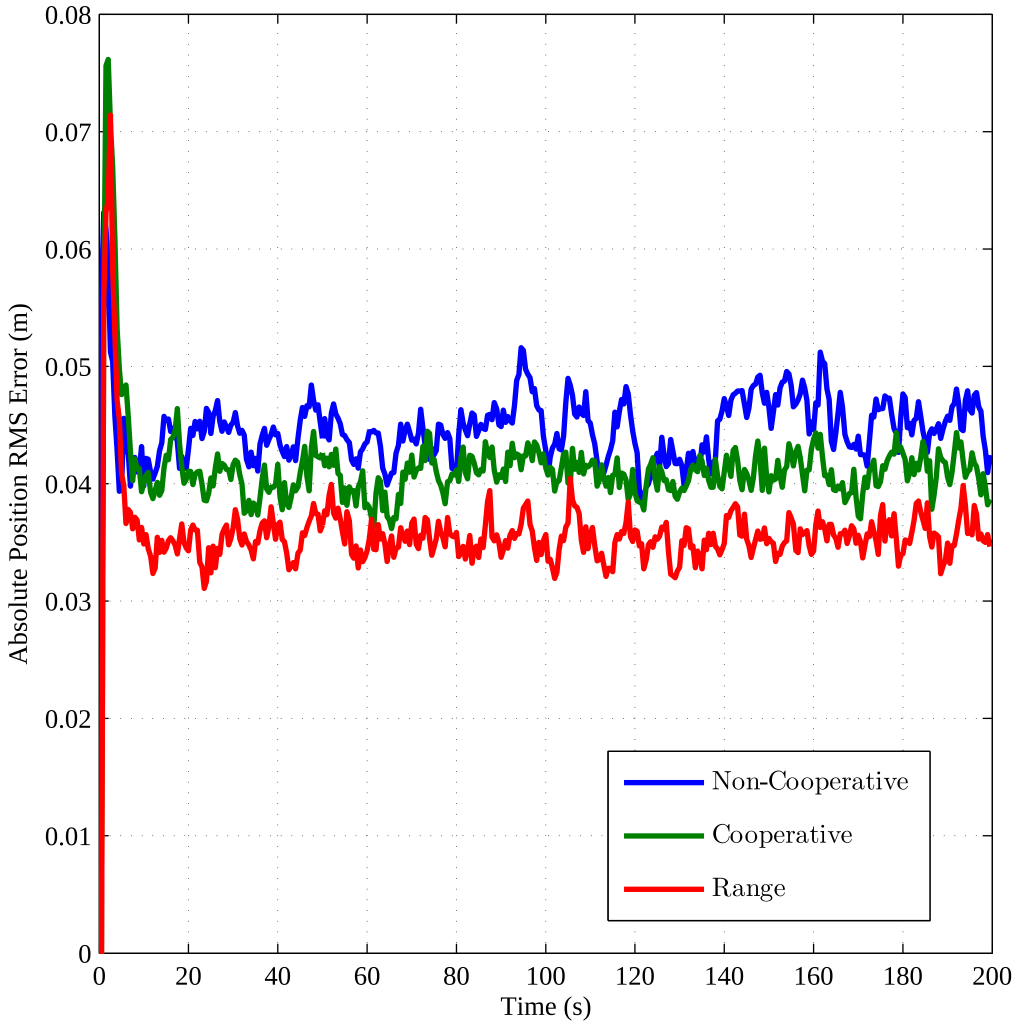


Figure 4.25: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute position error results. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

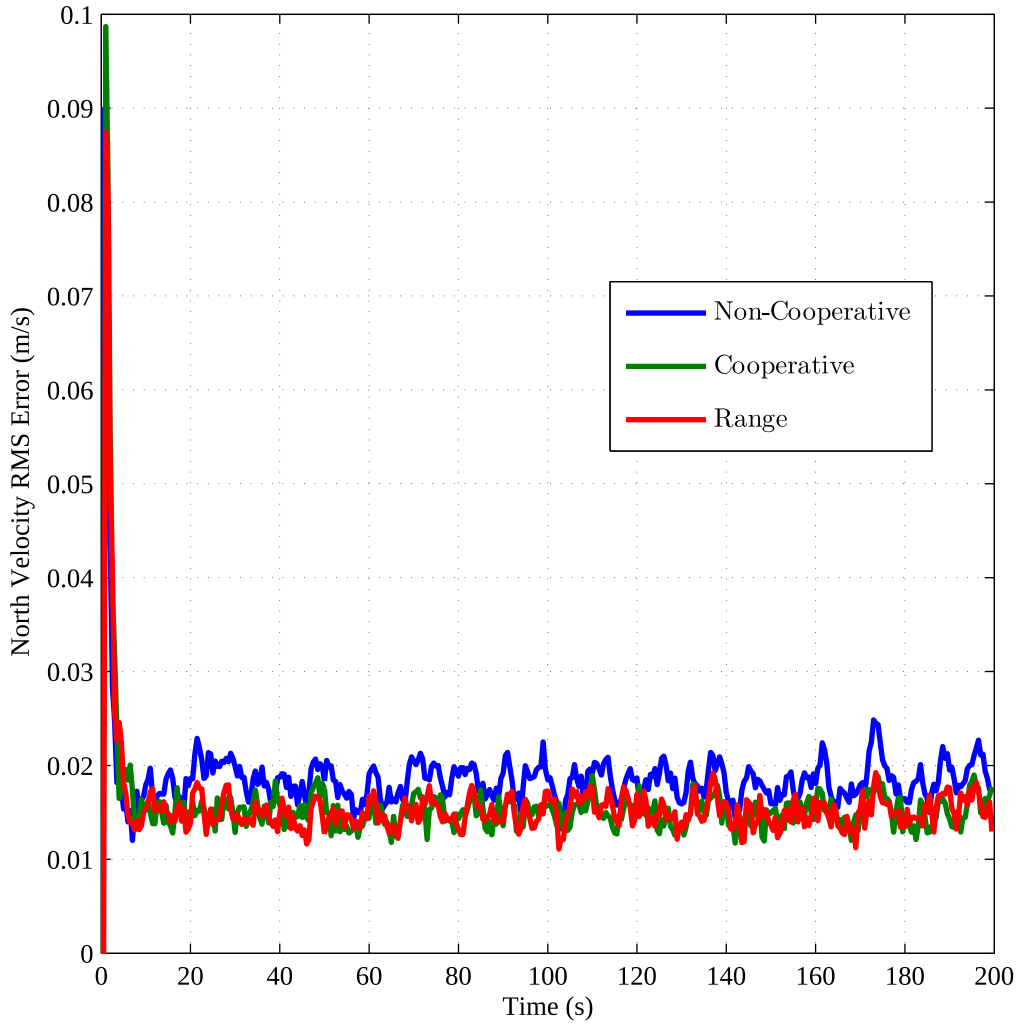


Figure 4.26: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

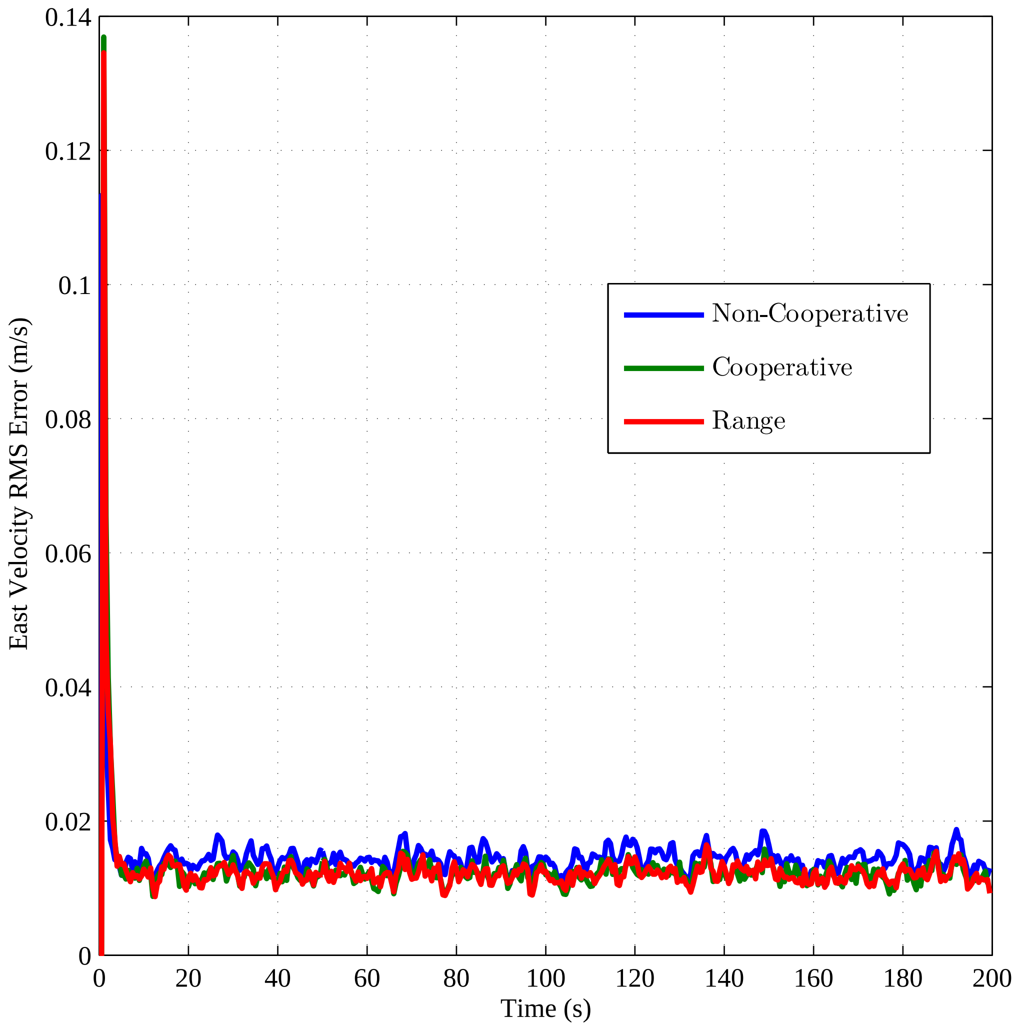


Figure 4.27: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

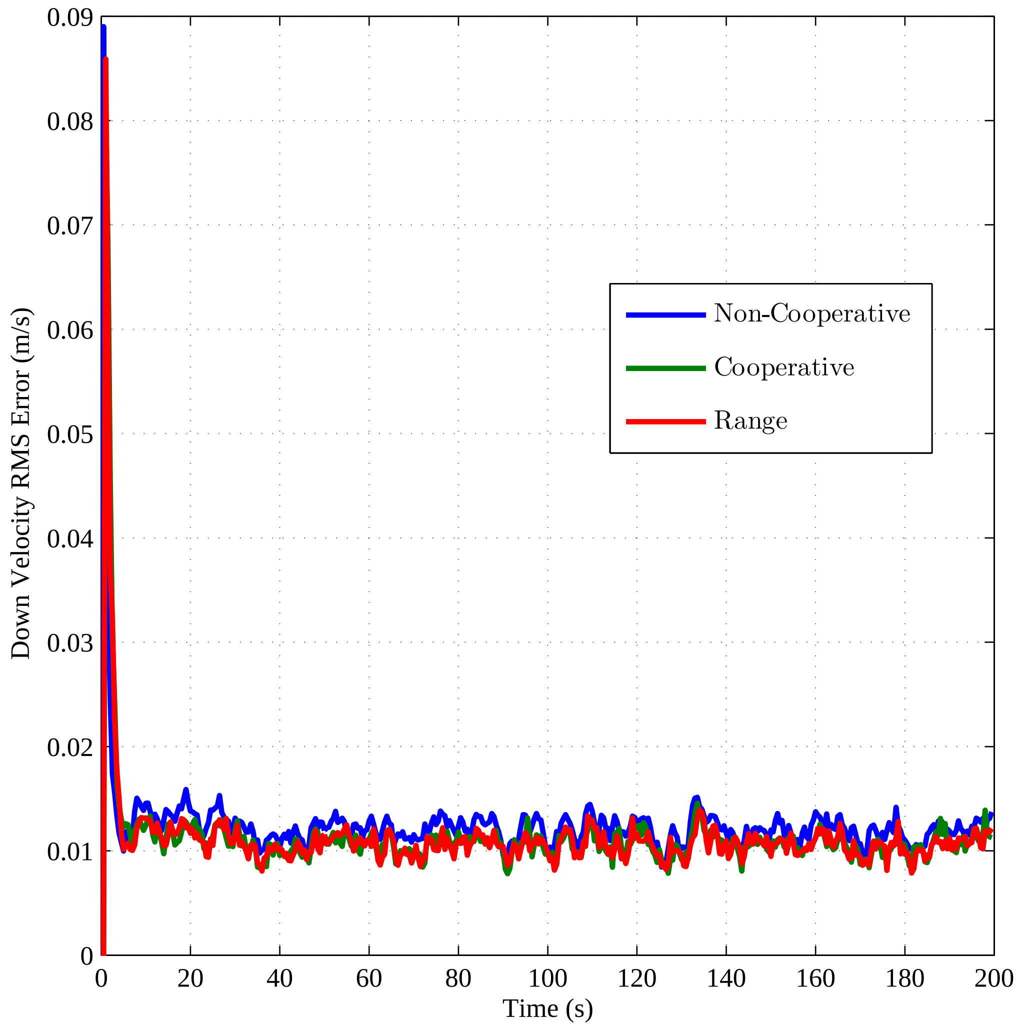


Figure 4.28: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

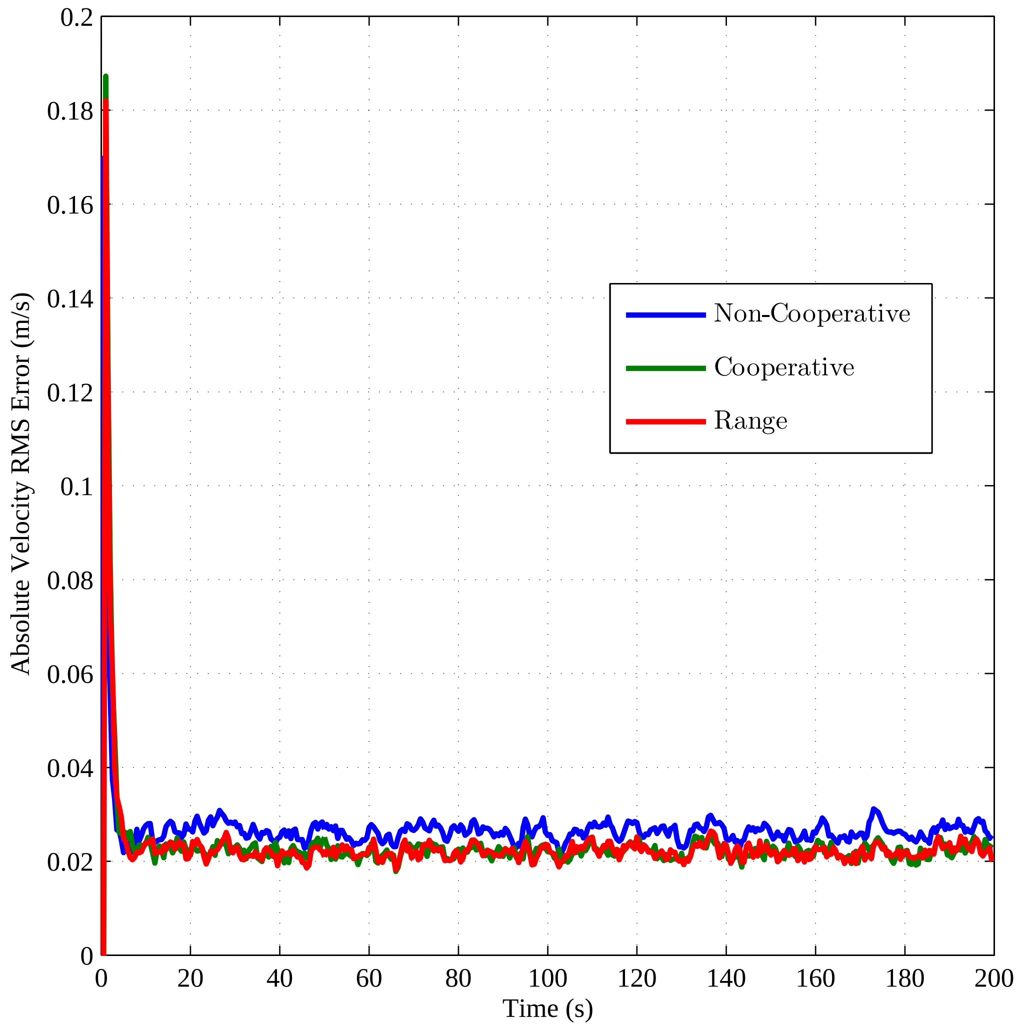


Figure 4.29: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute velocity error results . The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

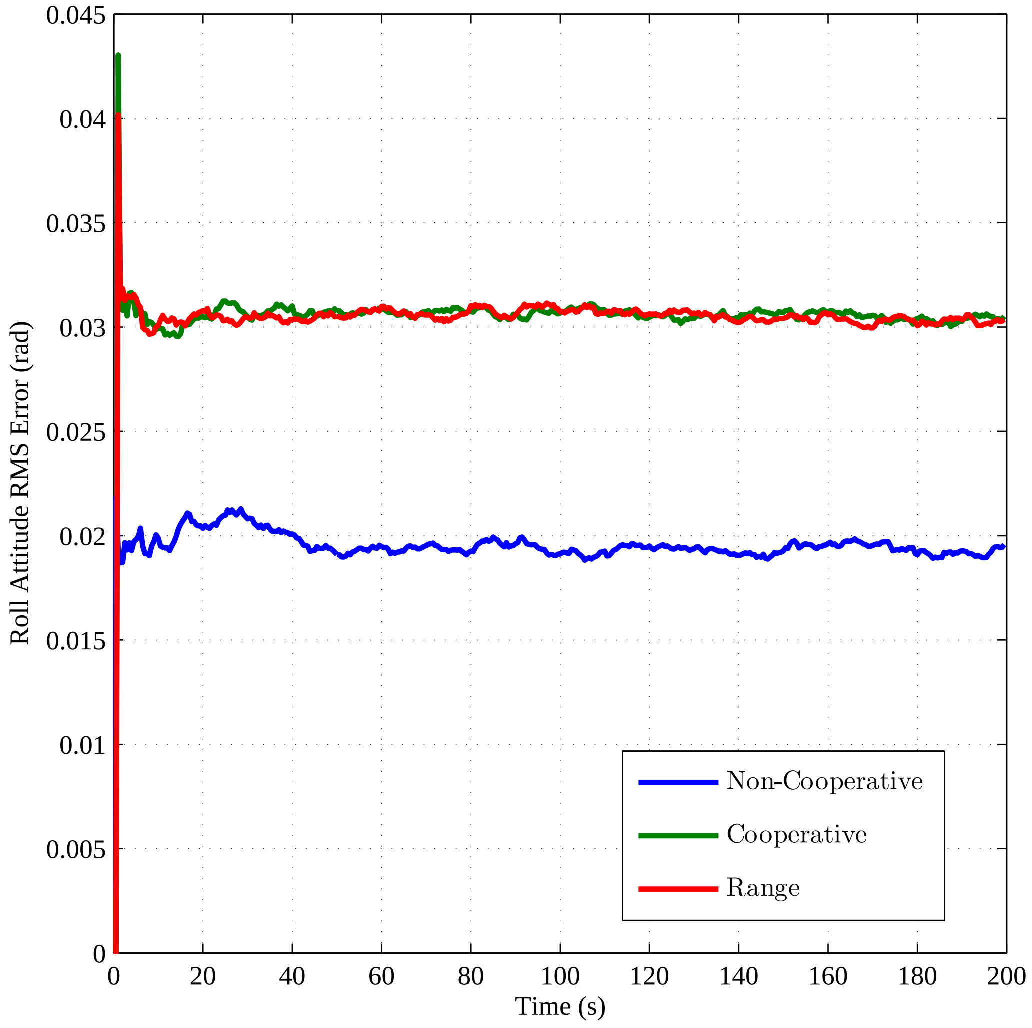


Figure 4.30: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

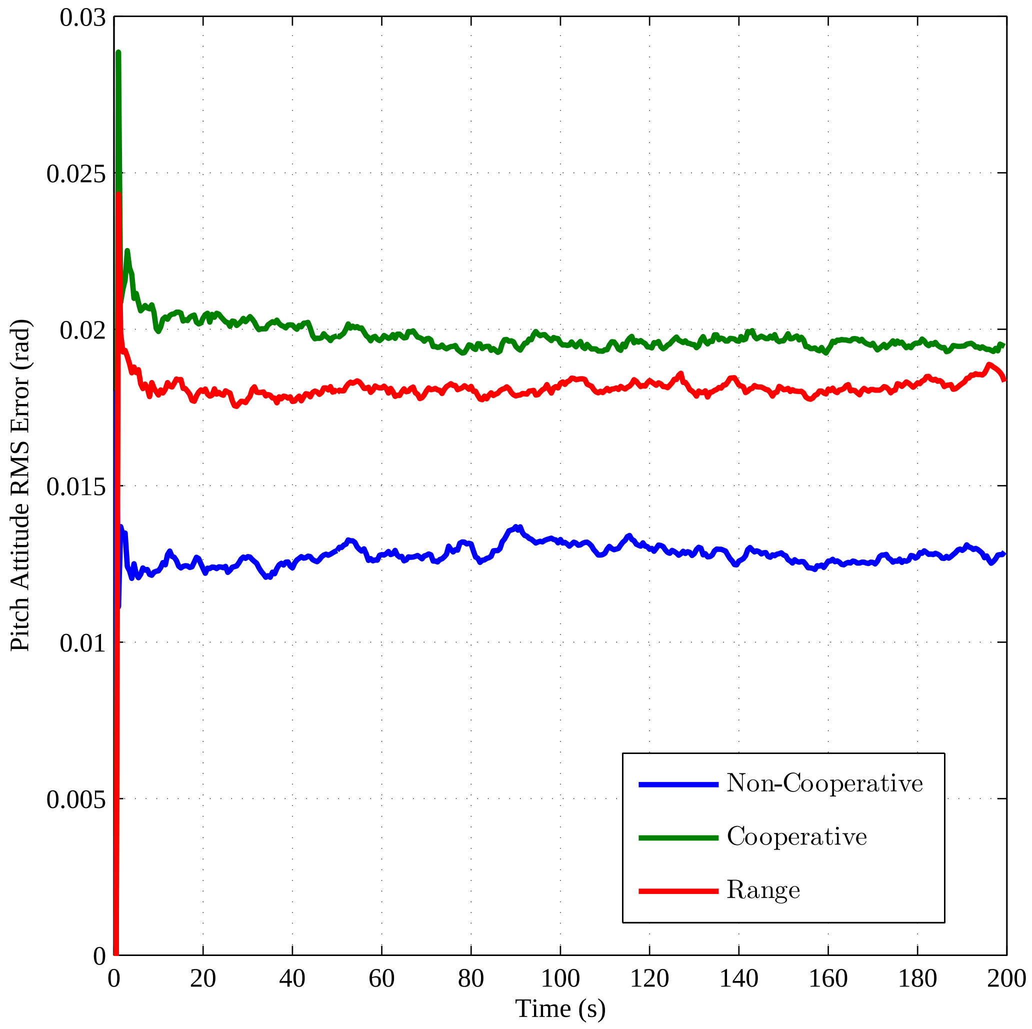


Figure 4.31: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

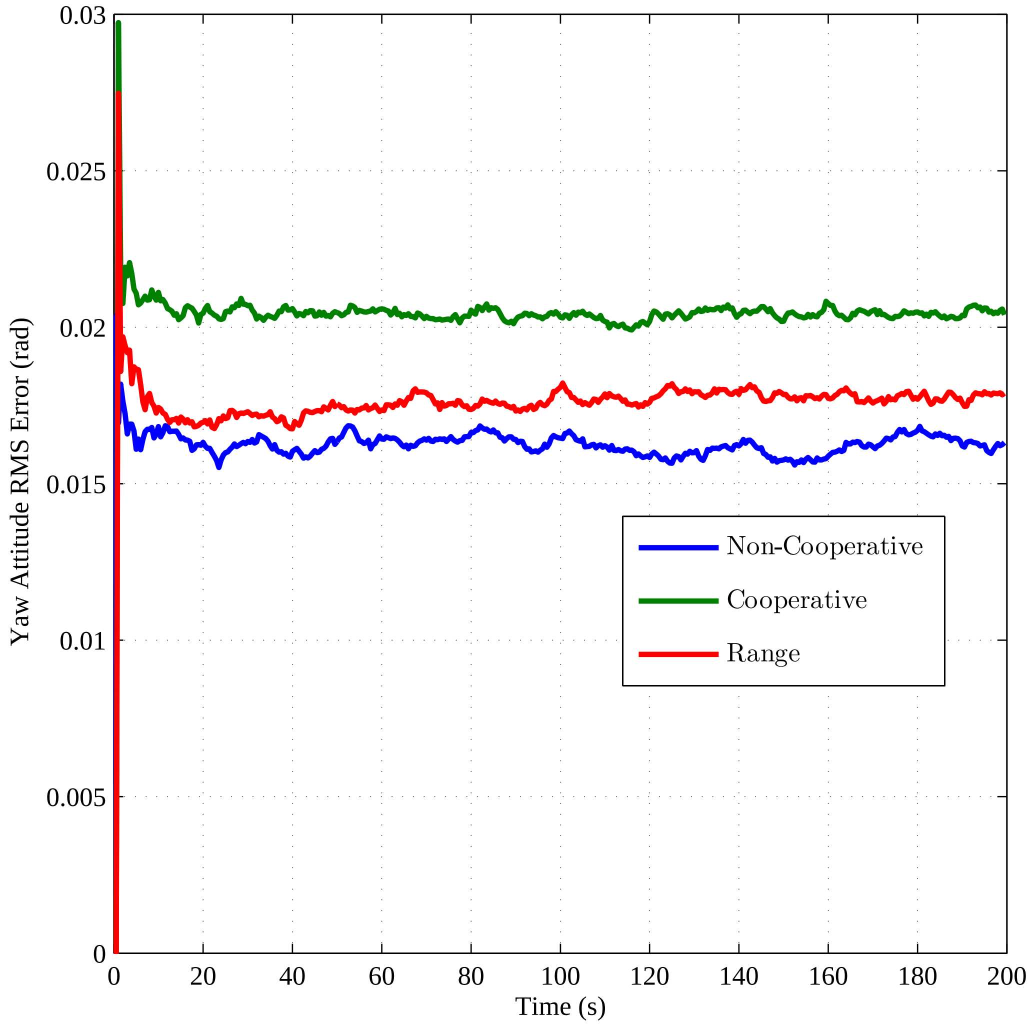


Figure 4.32: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

4.3.2 Hallway. The second simulation, using the pre-scaling algorithm, is designed to simulate a real-world environment and consists of two platforms traversing $15\,m$ of a $25\,m$ long simulated hallway. The hallway is $2\,m$ wide and $3\,m$ tall beginning $5\,m$ south of the *n-frame* origin and and ending $20\,m$ north of the origin. Eighty features are randomly placed along the walls, floor and ceiling of the hallway.

The first platform starts at the coordinate $[0, -0.5, 0.5]^T$, while the second platform starts at $[0, 0.5, -0.5]^T$. Both platforms hold these positions for 80 seconds and then accelerate to the north $0.025\,m/s$ for 15 seconds. Then, at 120 seconds the platforms accelerate $0.025\,m/s$ to the south for another 15 seconds ending at $[15, -0.5, 0.5]^T$ and $[15, 0.5, -0.5]^T$ respectively. The platforms then hold these positions for the remainder of the 200 second simulation. Figure 4.33 depicts the simulation scenario.

The simulation results are shown in Figures 4.34-4.44. As with the stationary simulation, the attitude errors are increased by the cooperative systems. However, due to the motion of the platforms beginning at 80 seconds, errors in the north axis are cross coupled into the east and down axes. This results in error baises, shown in Figures 4.35 and 4.36, after the platforms traverse the hallway.

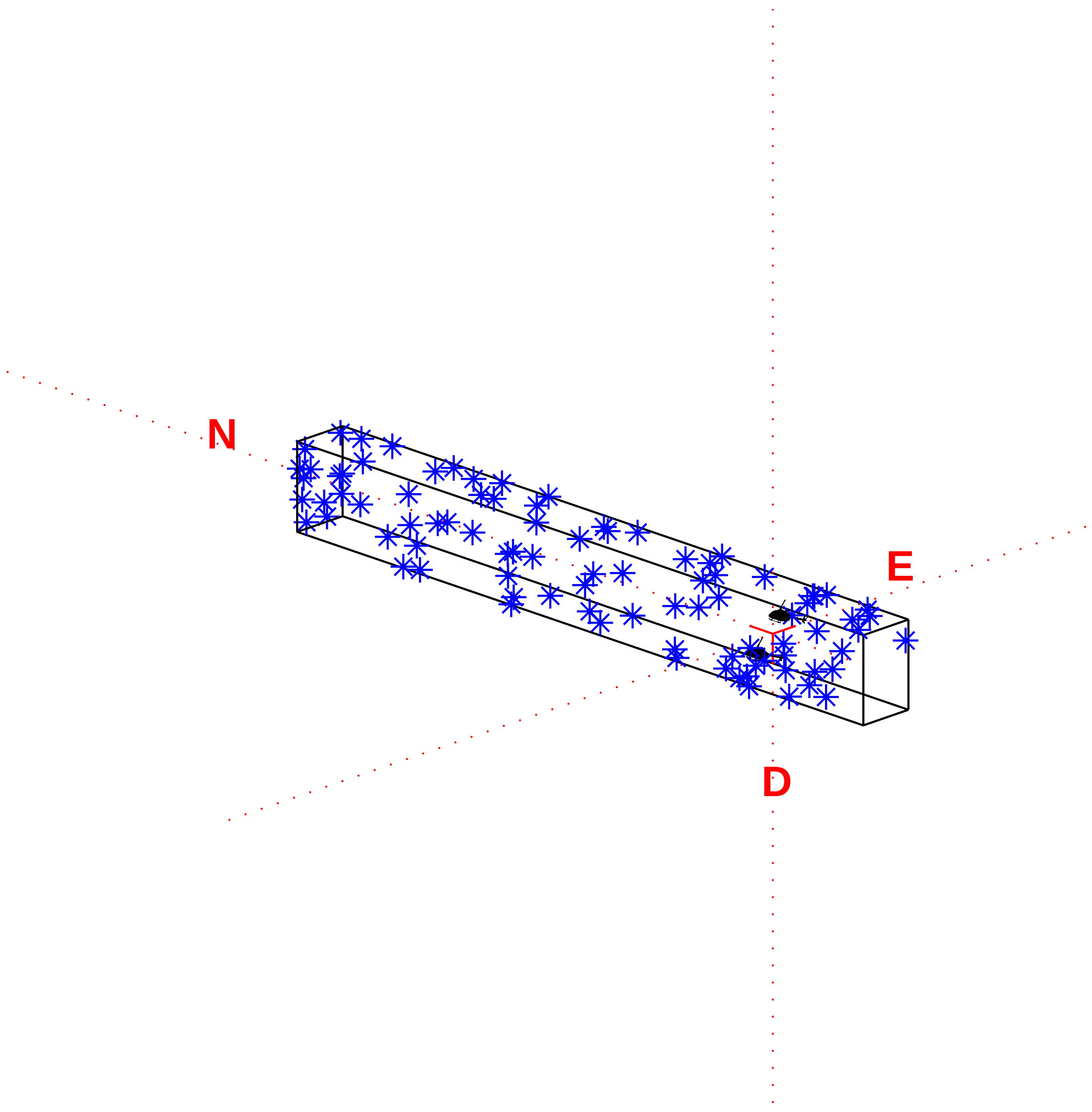


Figure 4.33: Two platform moving platform simulation scenario using the pre-scaling method. The platforms each travel (black lines) $15\,m$ along a $2\,m$ wide, $3\,m$ tall, and $25\,m$ long hallway. Eighty random features (blue asterisk) are positioned randomly on the walls, floor, and ceiling of the hallway.

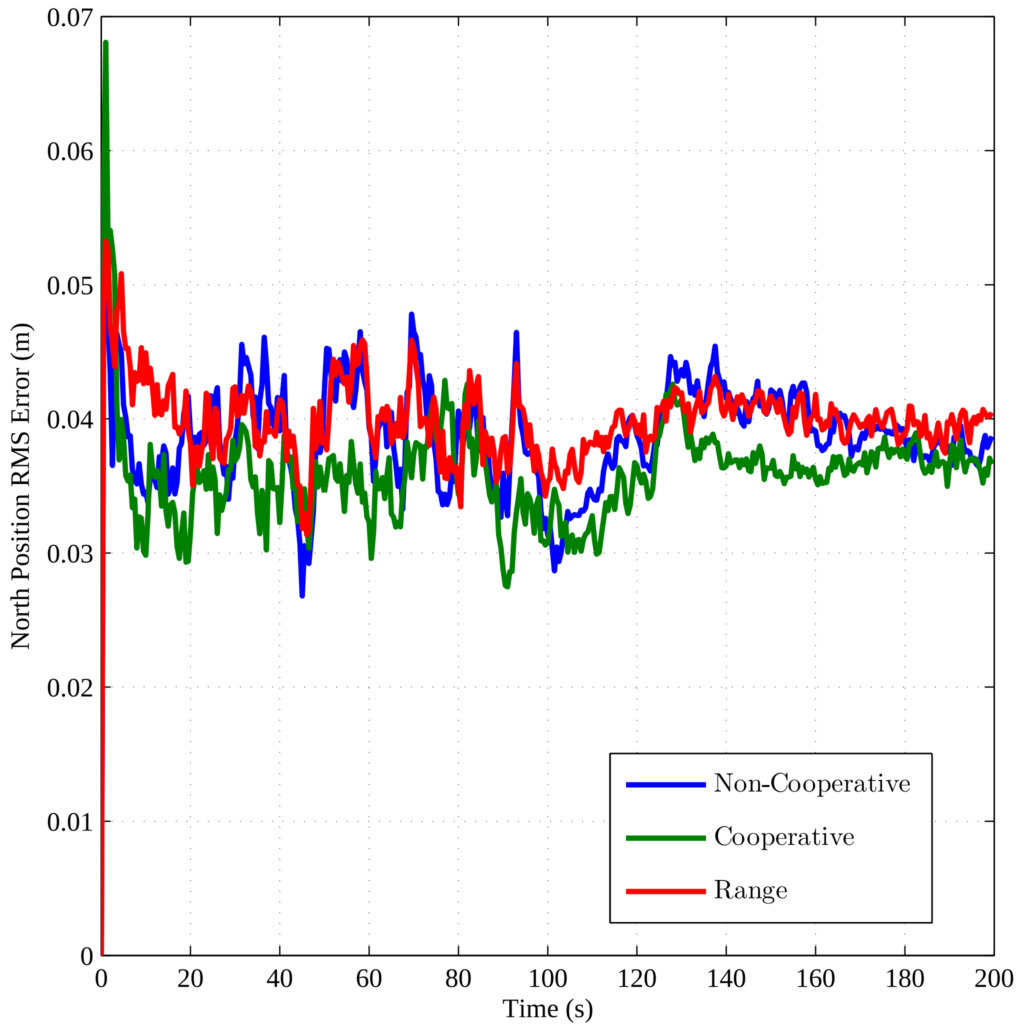


Figure 4.34: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a $25\,m$ hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

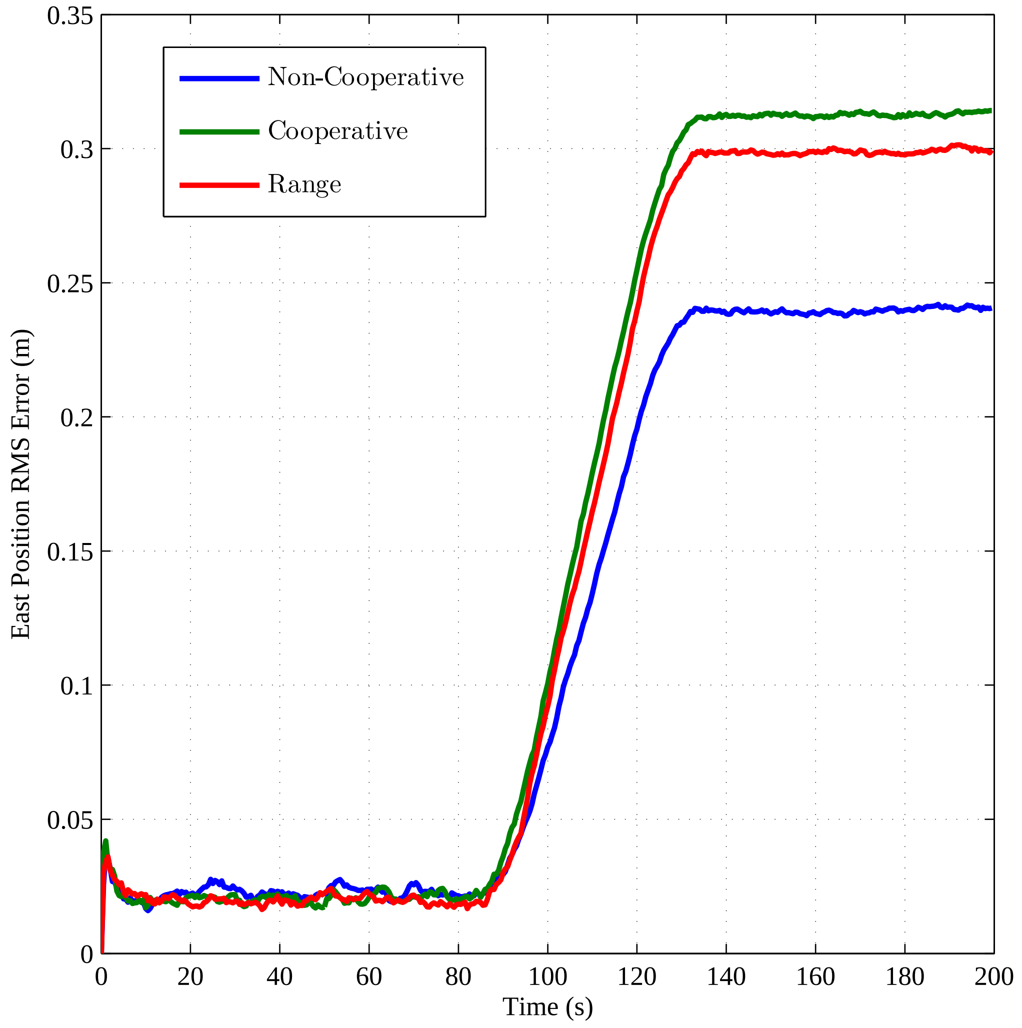


Figure 4.35: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the east axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a $25\,m$ hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

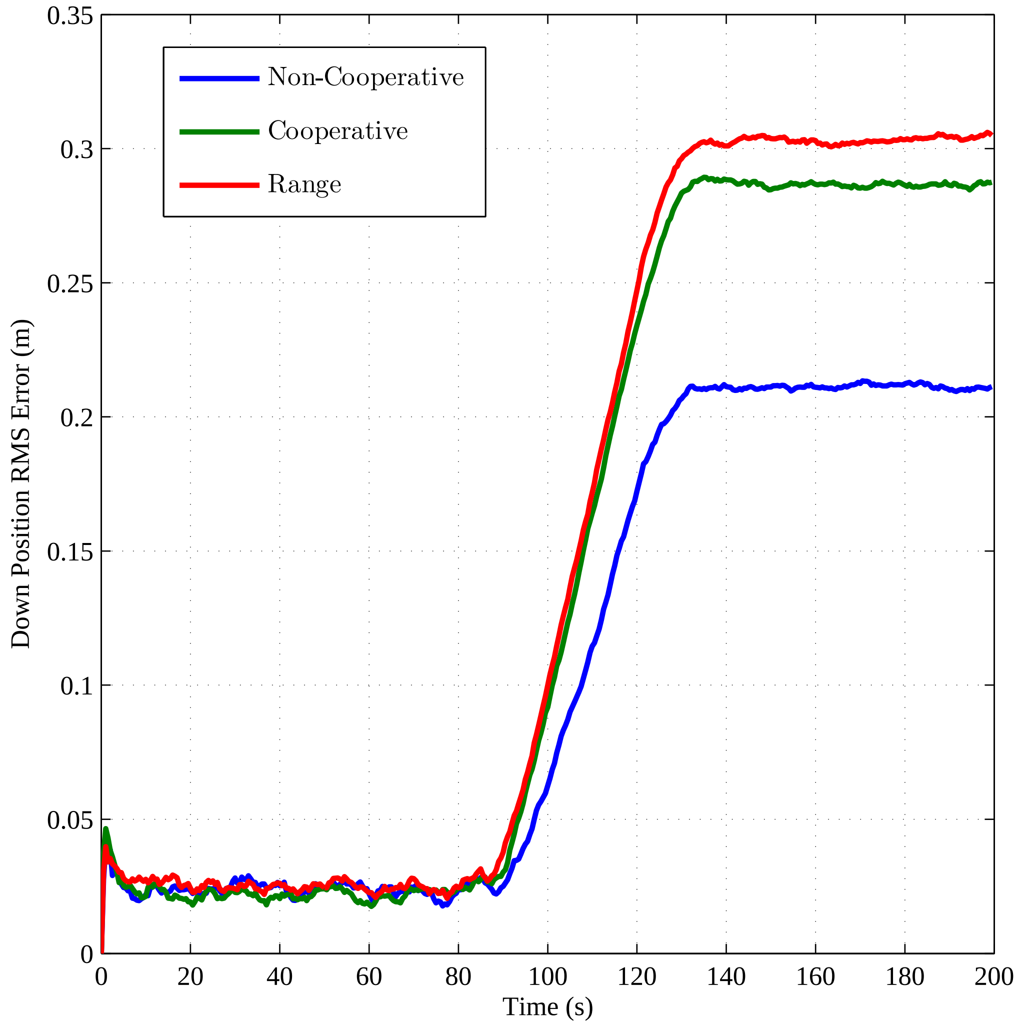


Figure 4.36: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

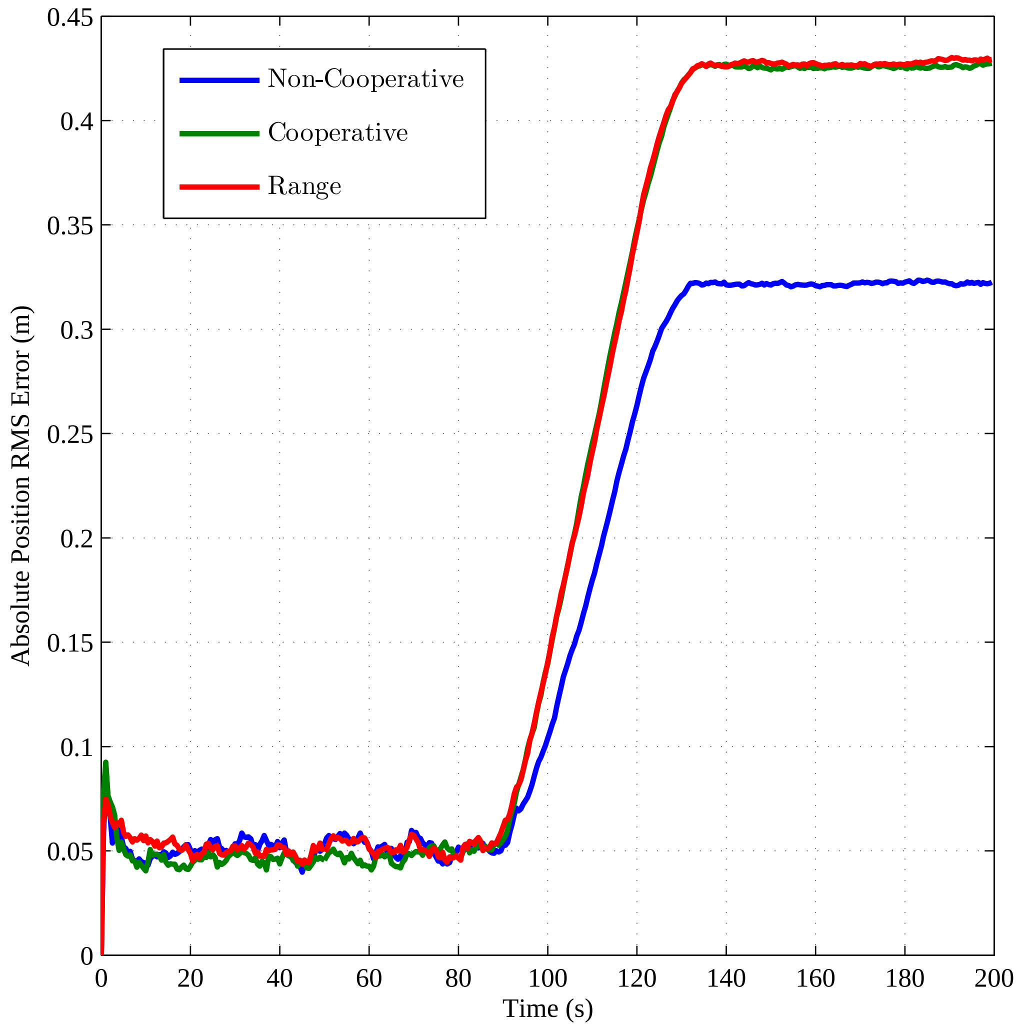


Figure 4.37: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute position error results. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the prescaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

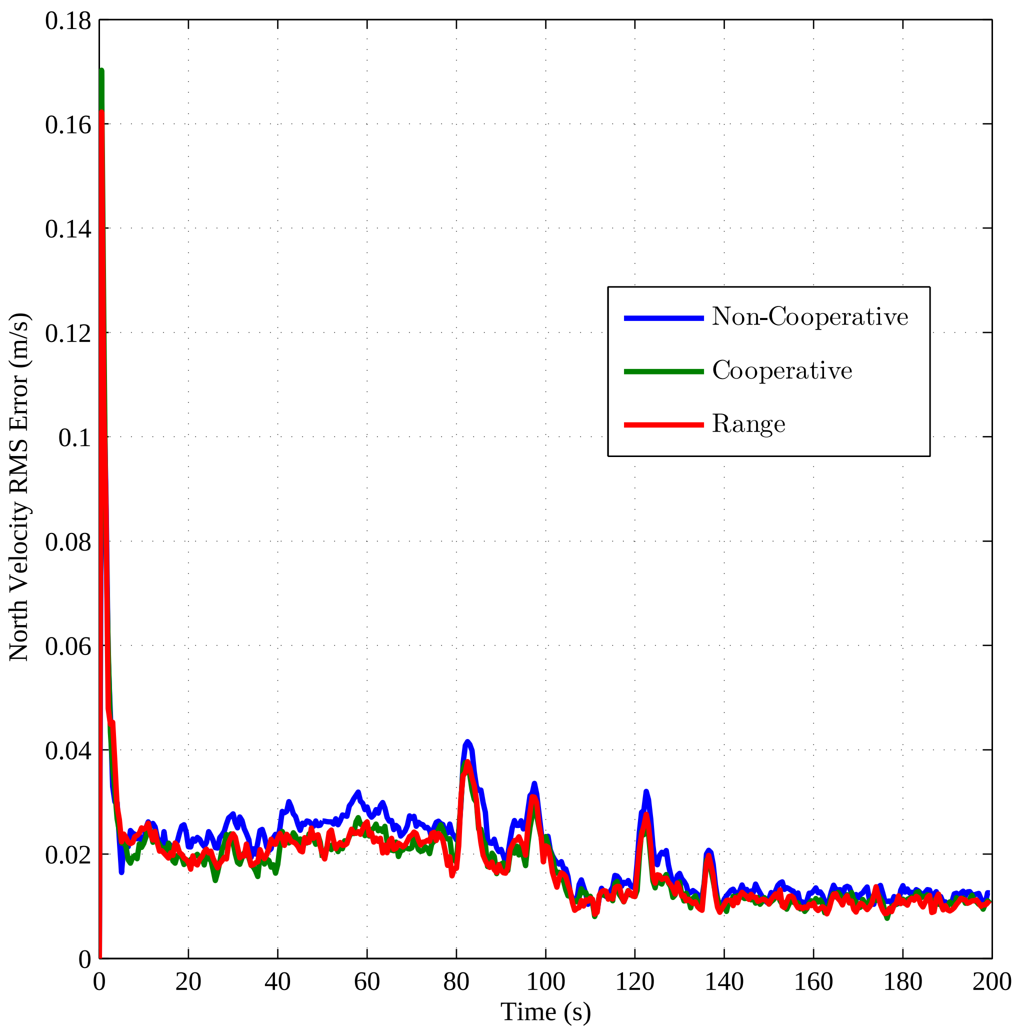


Figure 4.38: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

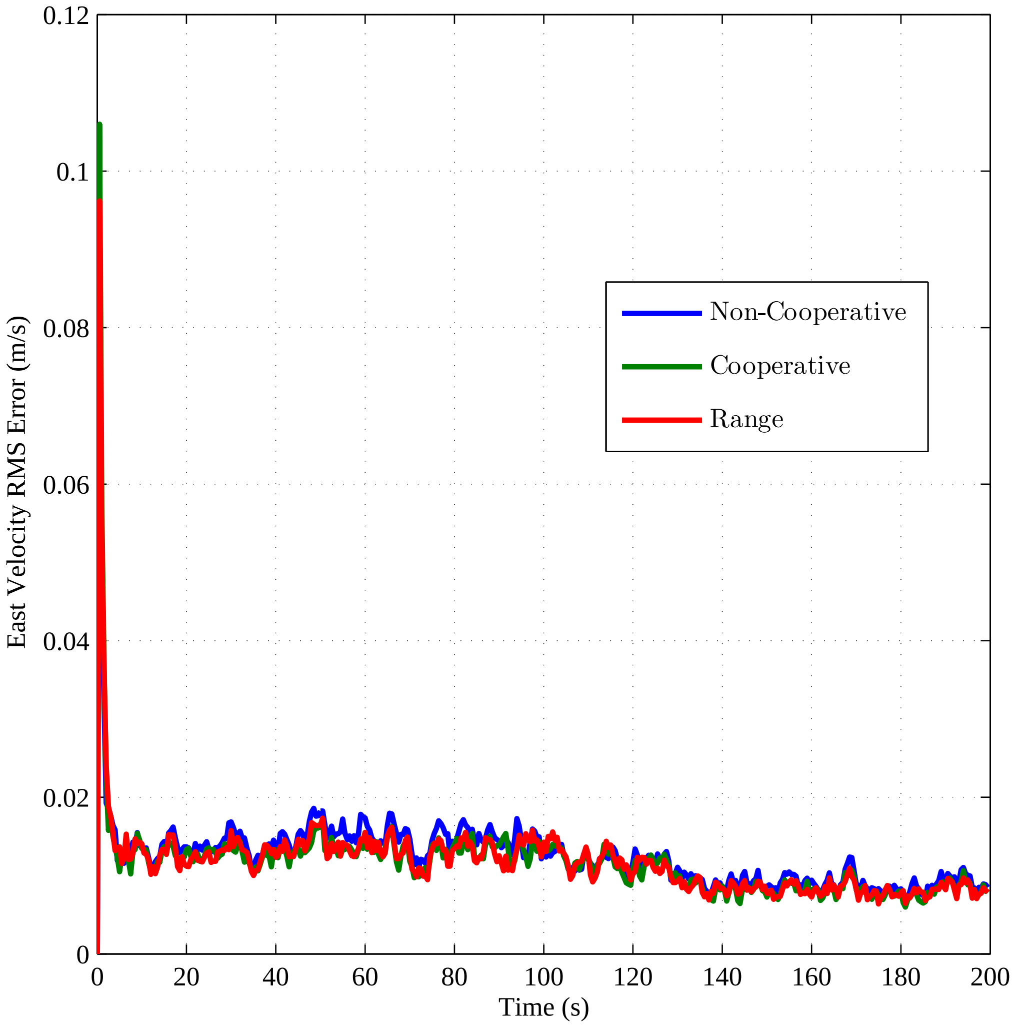


Figure 4.39: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the east axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a $25\,m$ hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

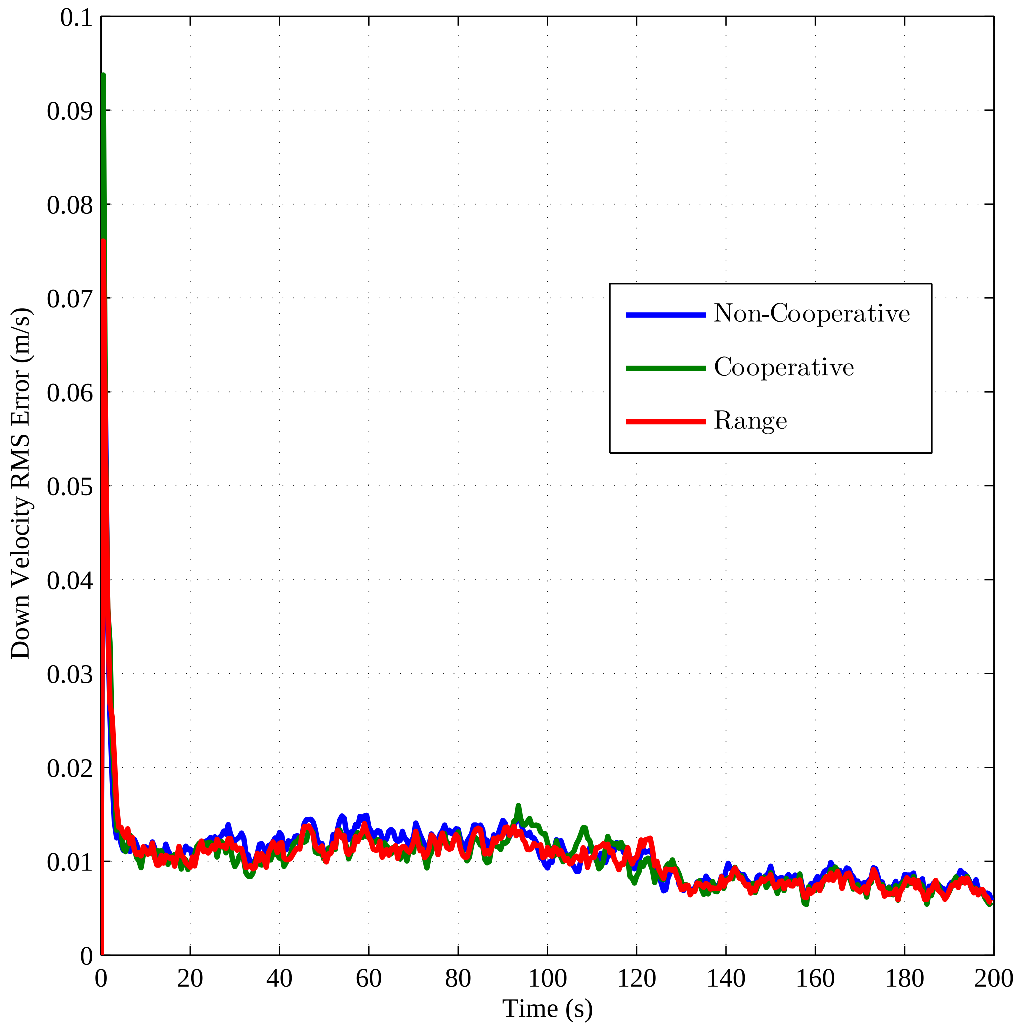


Figure 4.40: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

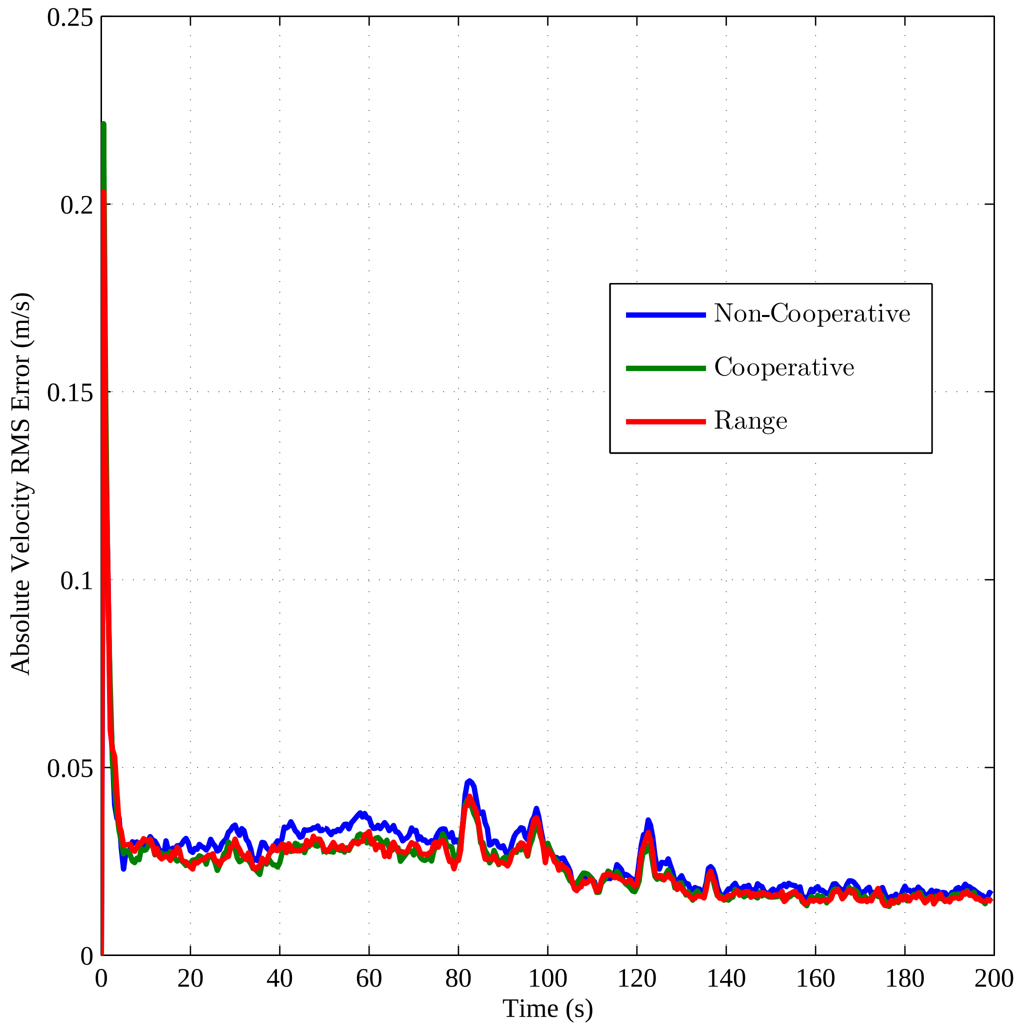


Figure 4.41: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute velocity error results . The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the prescaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

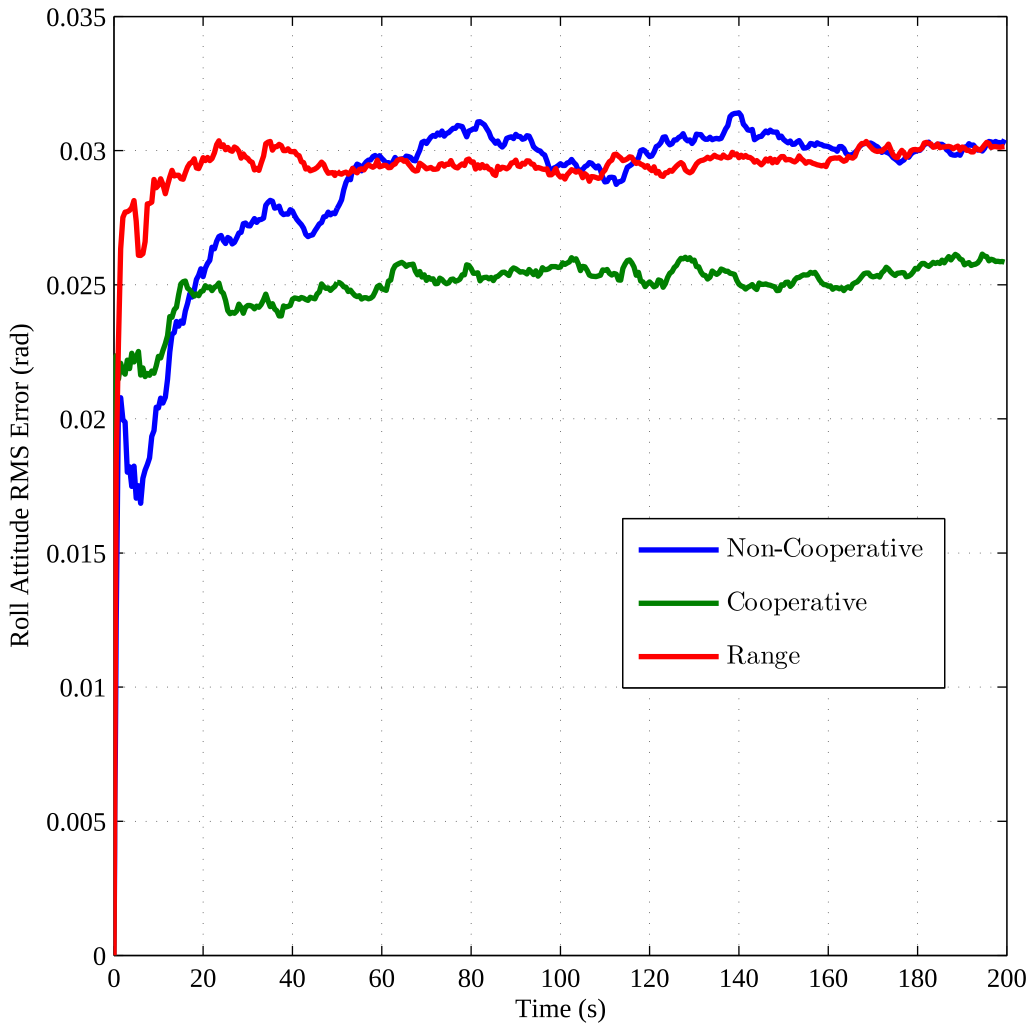


Figure 4.42: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

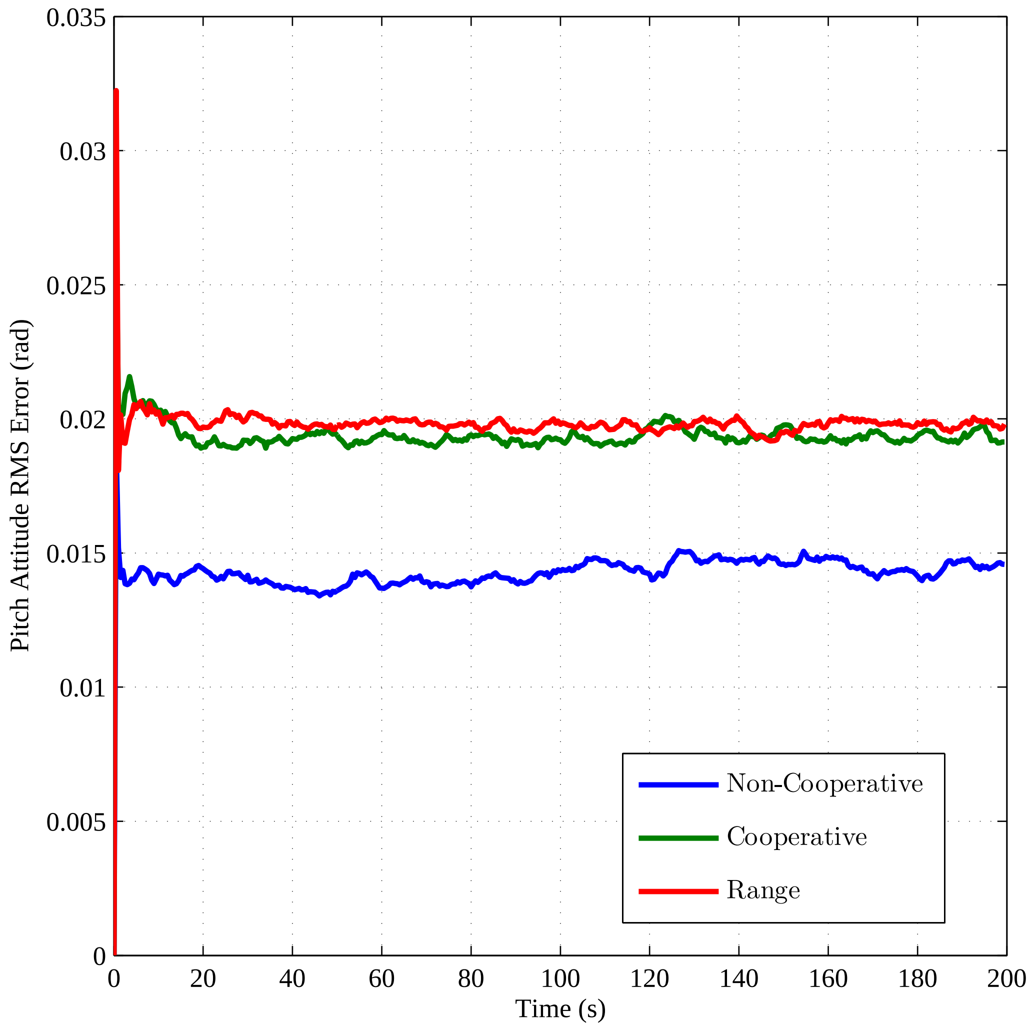


Figure 4.43: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a $25\,m$ hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

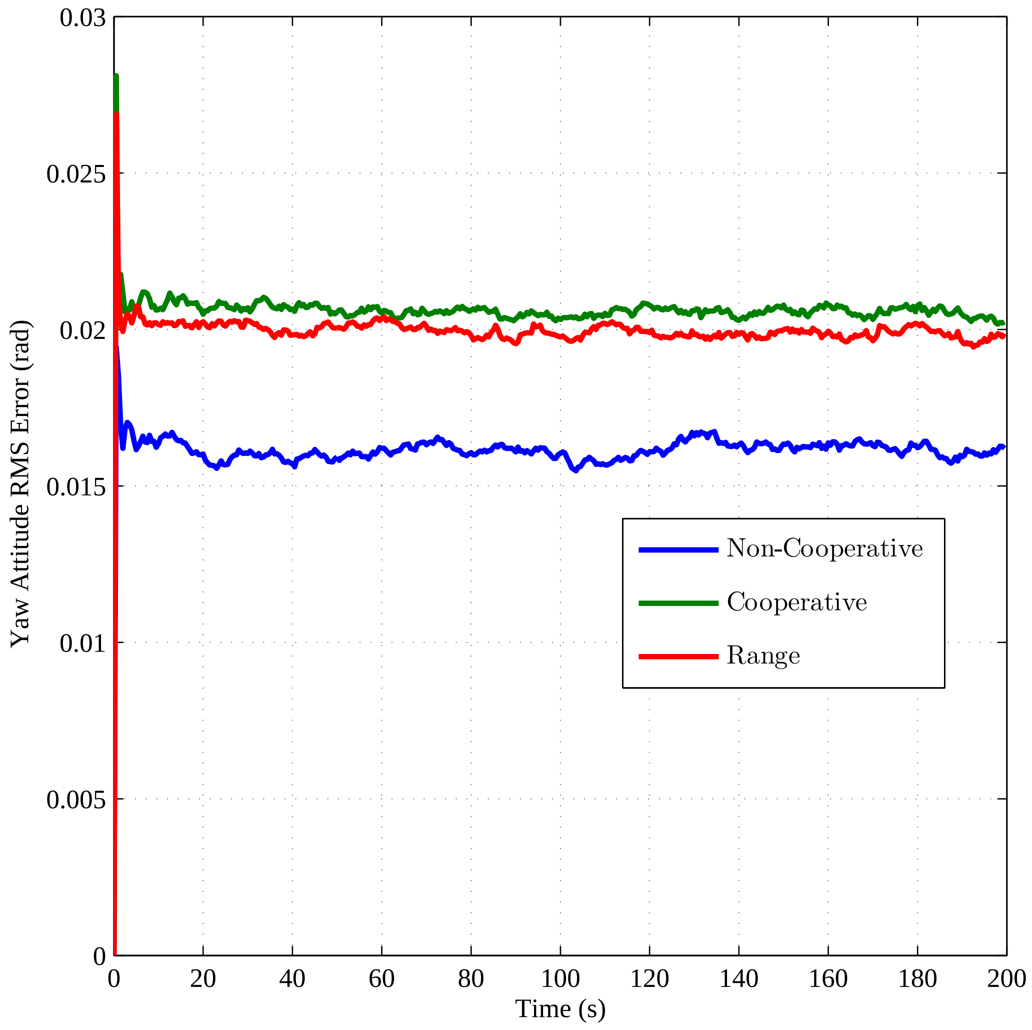


Figure 4.44: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

4.3.3 Moving Platforms. The third simulation, using the pre-scaling algorithm, tests the cooperative systems' abilities to compute solutions of moving platforms. The two platforms perform sideslip orbits of the reference feature cloud. The sideslip orbit is characterized by the orientation of the platform as it circles a point in space. At all times the platform is oriented so the x-axis of the body frame of reference is pointed at the center point of the orbit.

The first platform begins and ends the sideslip orbit $1\,m$ above the *n-frame* origin and orbits the point at $[10, 0, -1]^T$, counterclockwise, at a constant altitude of $1\,m$ above the *n-frame* origin. The second platform follows an identical orbital path 1m below the *n-frame* origin and starting $45°$ clockwise from the *n-frame* origin. Each platform completes one full orbit of the feature cloud over the course of the simulation.

The simulation utilizes 30 reference features distributed in the same manner as the stationary simulations performed previously. Figure 4.45 depicts the simulation scenario. The 50 run Monte Carlo simulation is performed over $200\,s$ at a $2\,Hz$ sampling rate.

The results of the sideslip simulation are shown in Figures 4.46-4.56. As expected, the position errors are reduced by the cooperative systems. The absolute position is improved by up to $10\,cm$ depending on the position of the platforms in the orbit.

This simulation demonstrates how changing the attitude and position, of the platforms, eliminates the attitude errors displayed by the stationary and hallway simulations. The reasoning behind this result is discussed in Section 4.3.5.

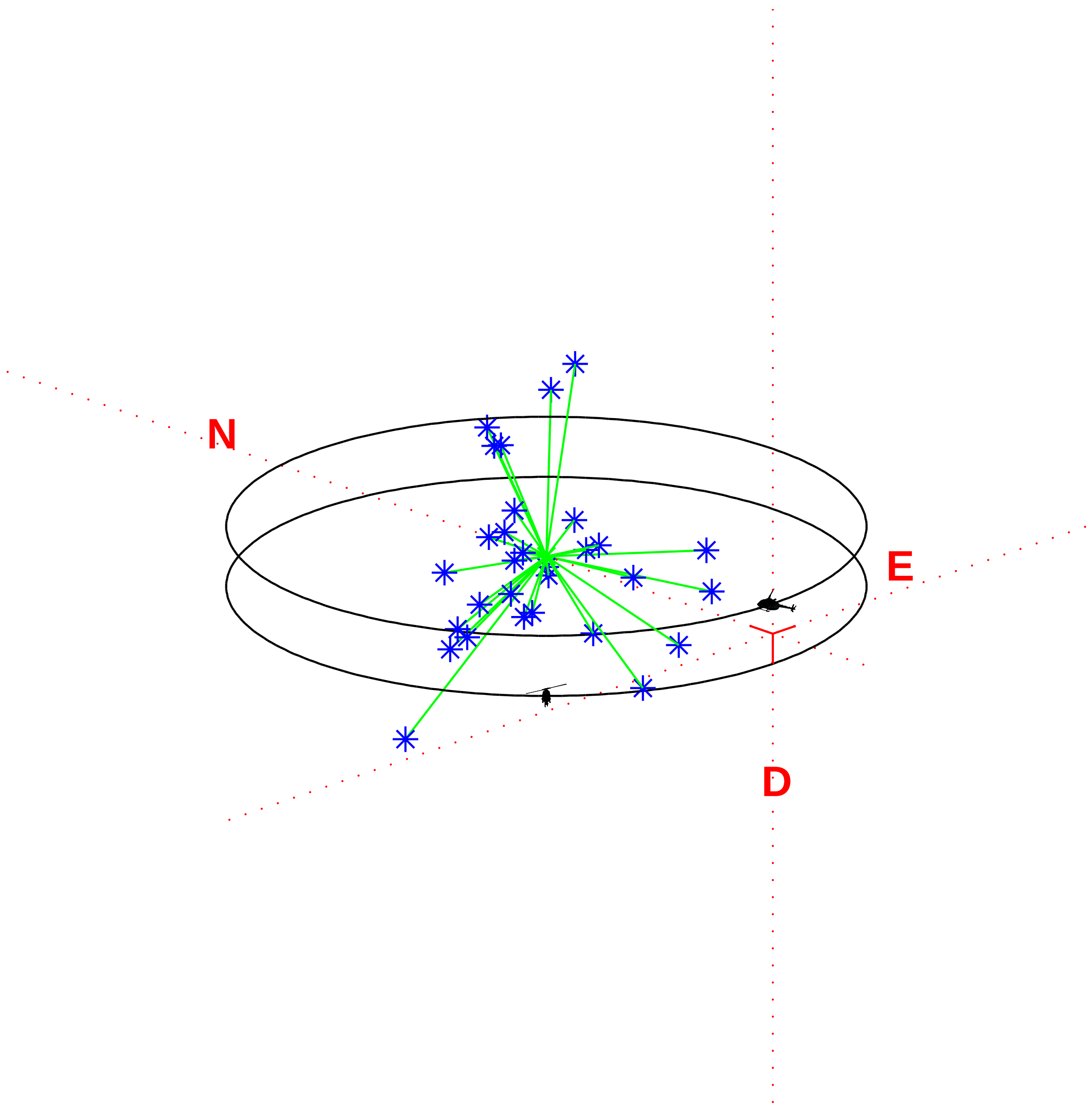


Figure 4.45: Two platform moving platform simulation scenario using the pre-scaling method. The platforms each perform a counterclockwise, sideslip orbit (black lines) of the central point of the feature cloud at a radius of $10\,m$. The two platforms are separated 45° around the orbit and 1 m above and below the local navigation frame x-y axis plane. Thirty random features (blue asterisk) are positioned around the coordinate $[10, 0, 0]^T$.

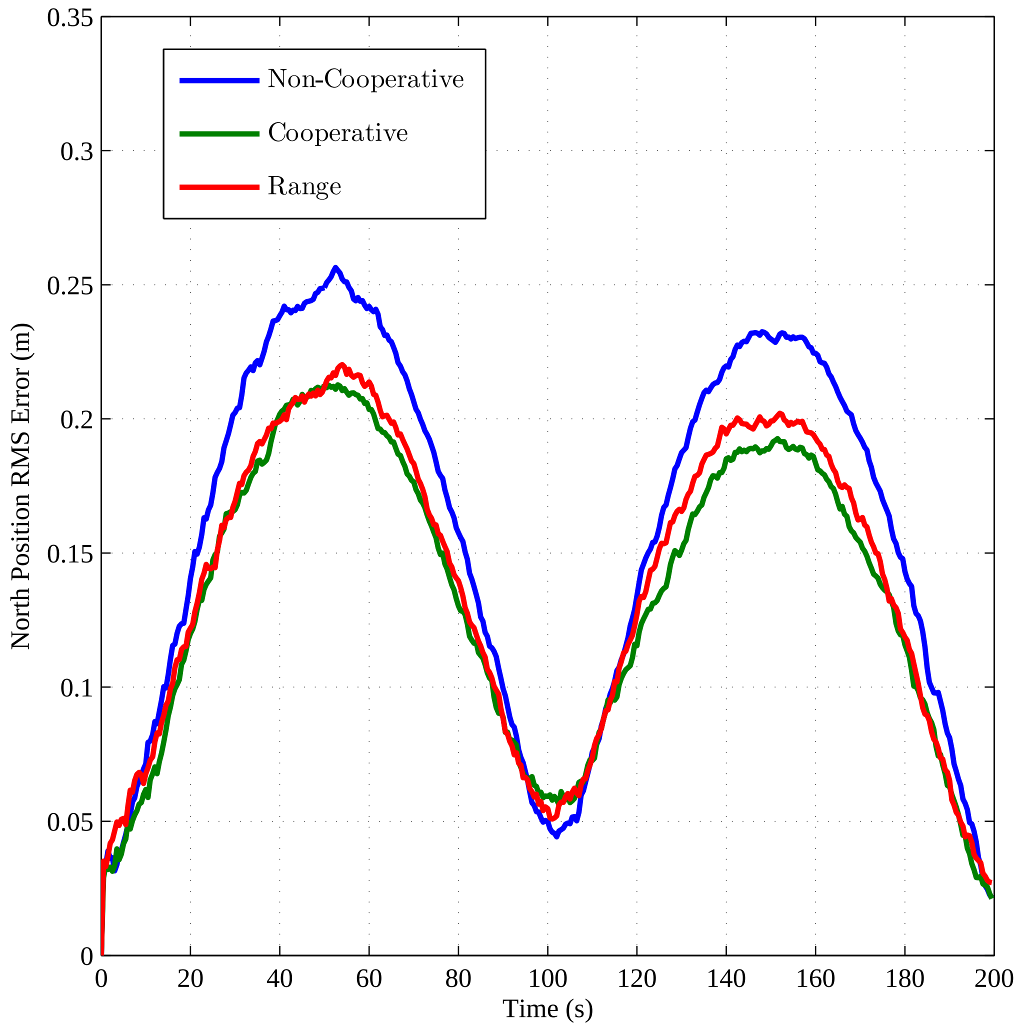


Figure 4.46: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

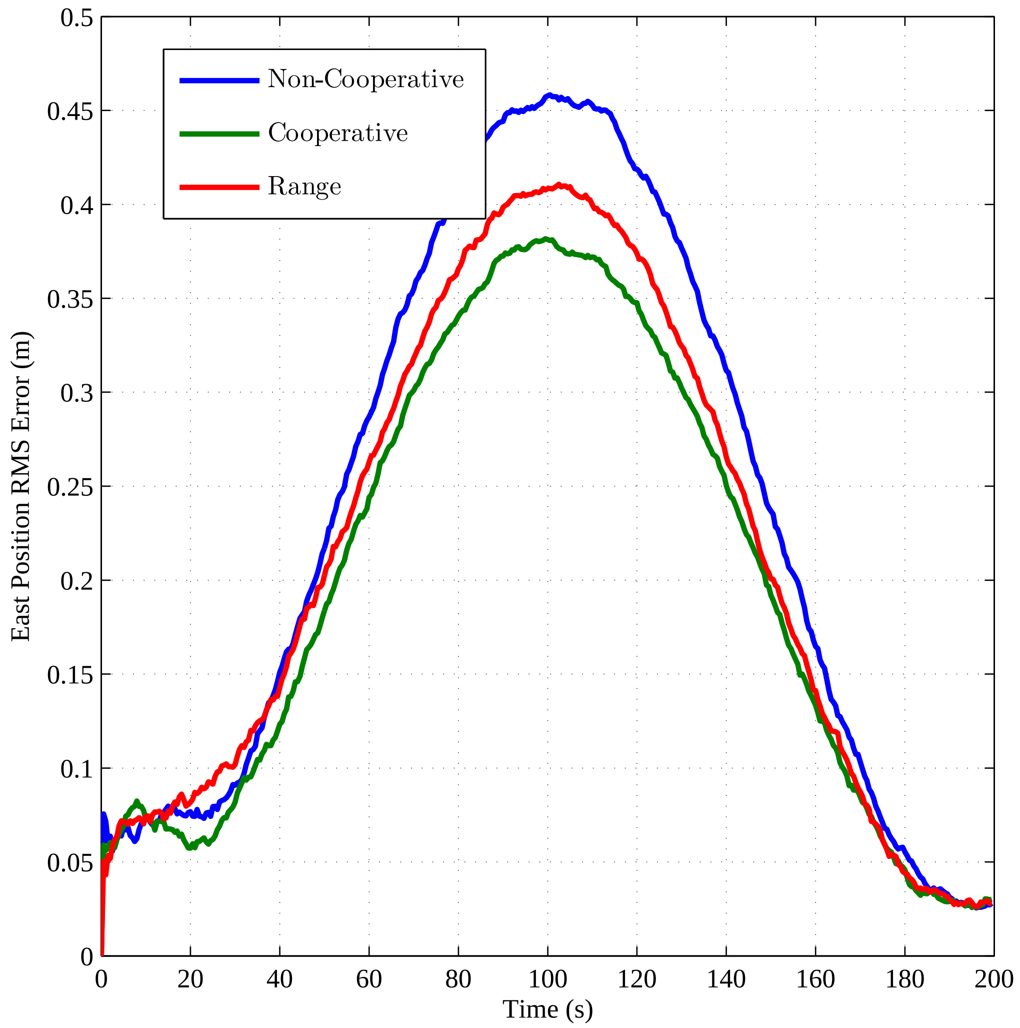


Figure 4.47: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the east axis of the local navigation frame. The results shown are for the first of two platforms traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

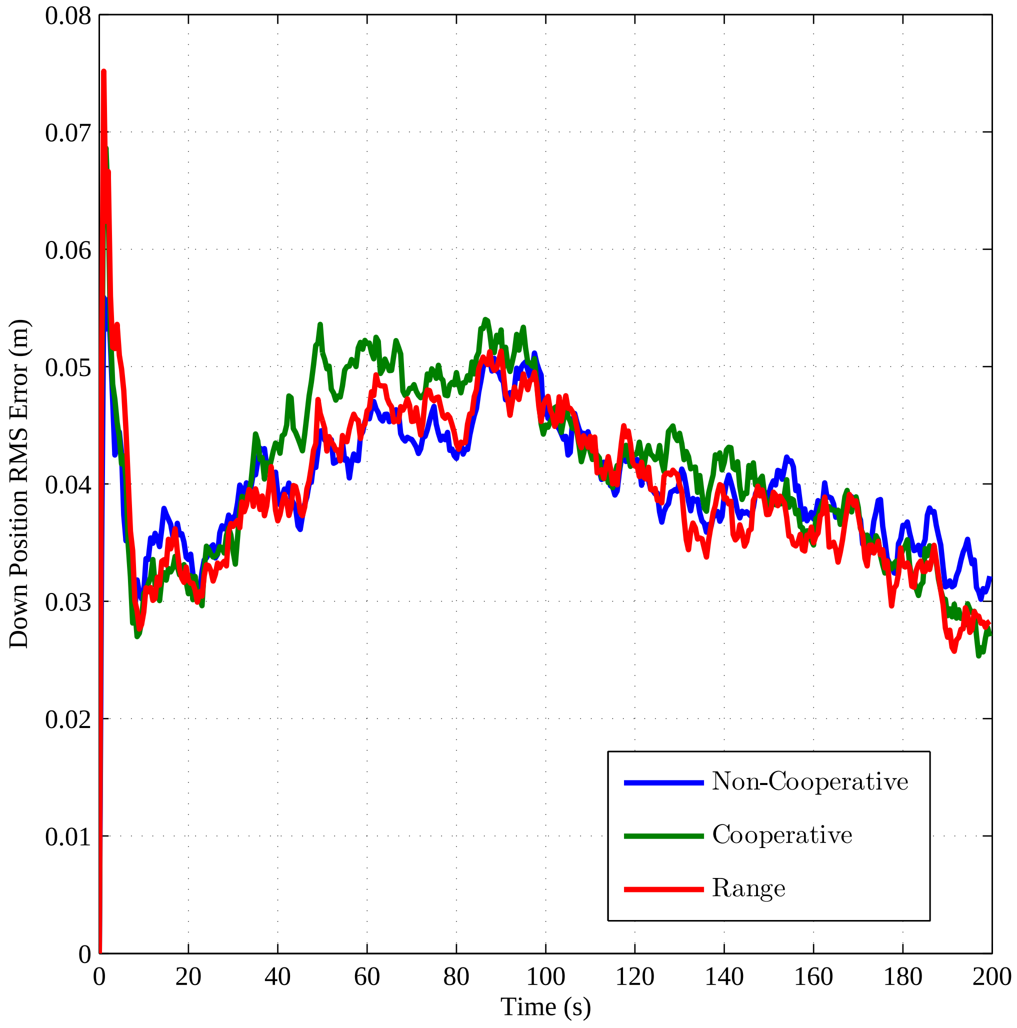


Figure 4.48: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

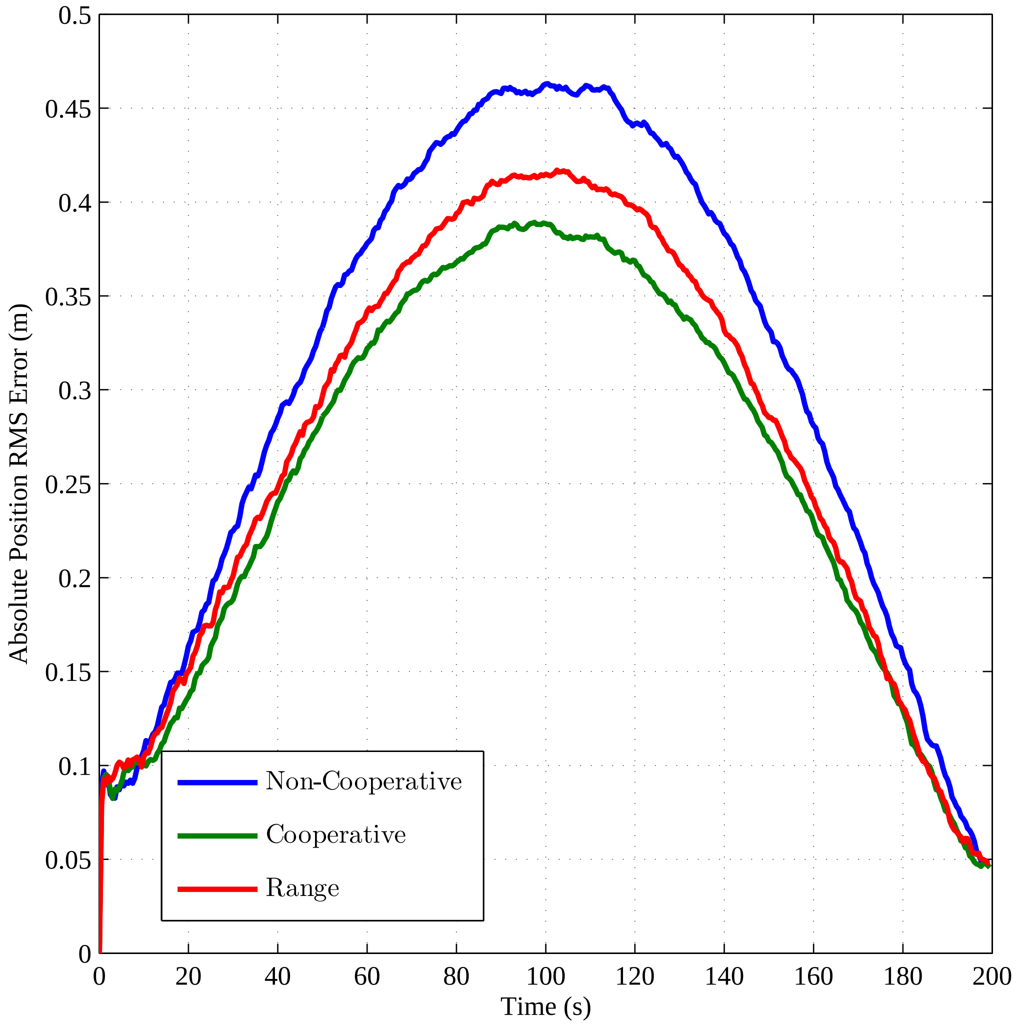


Figure 4.49: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute position error results. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

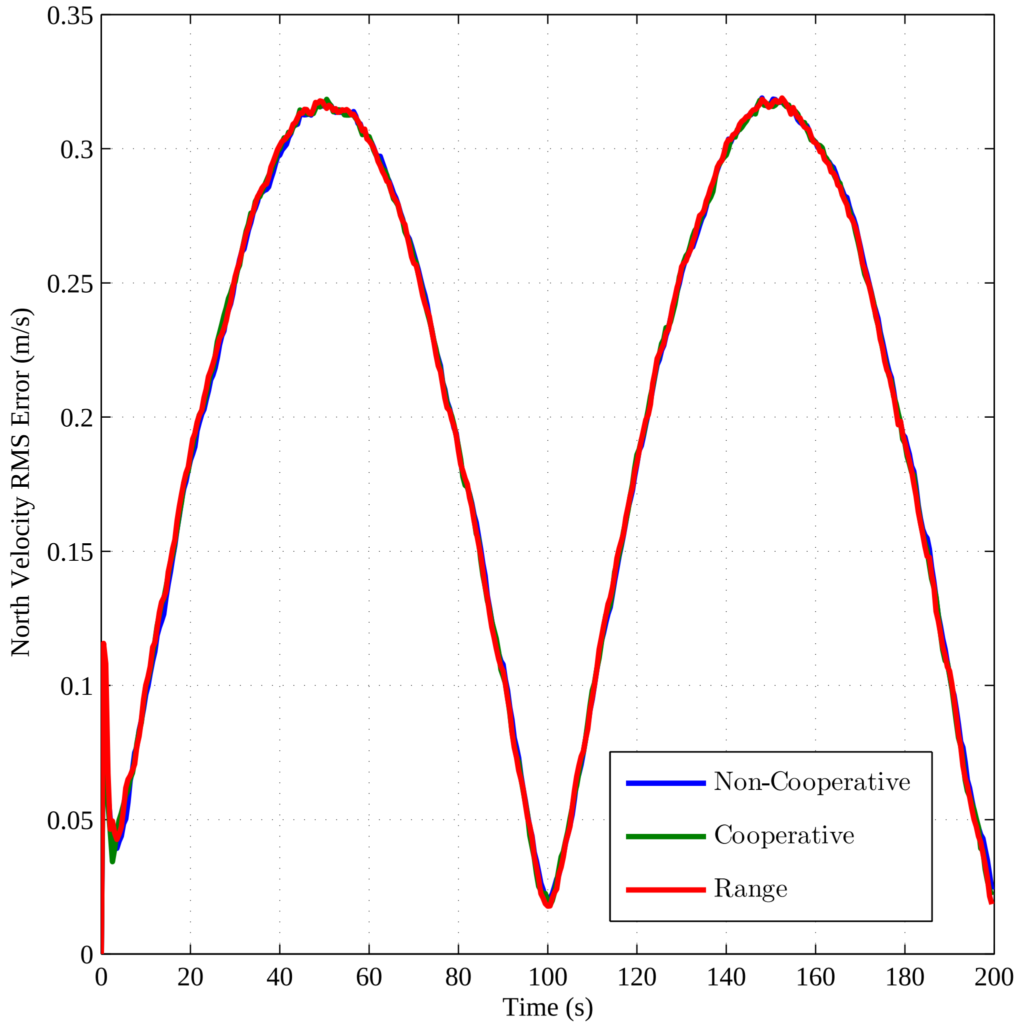


Figure 4.50: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

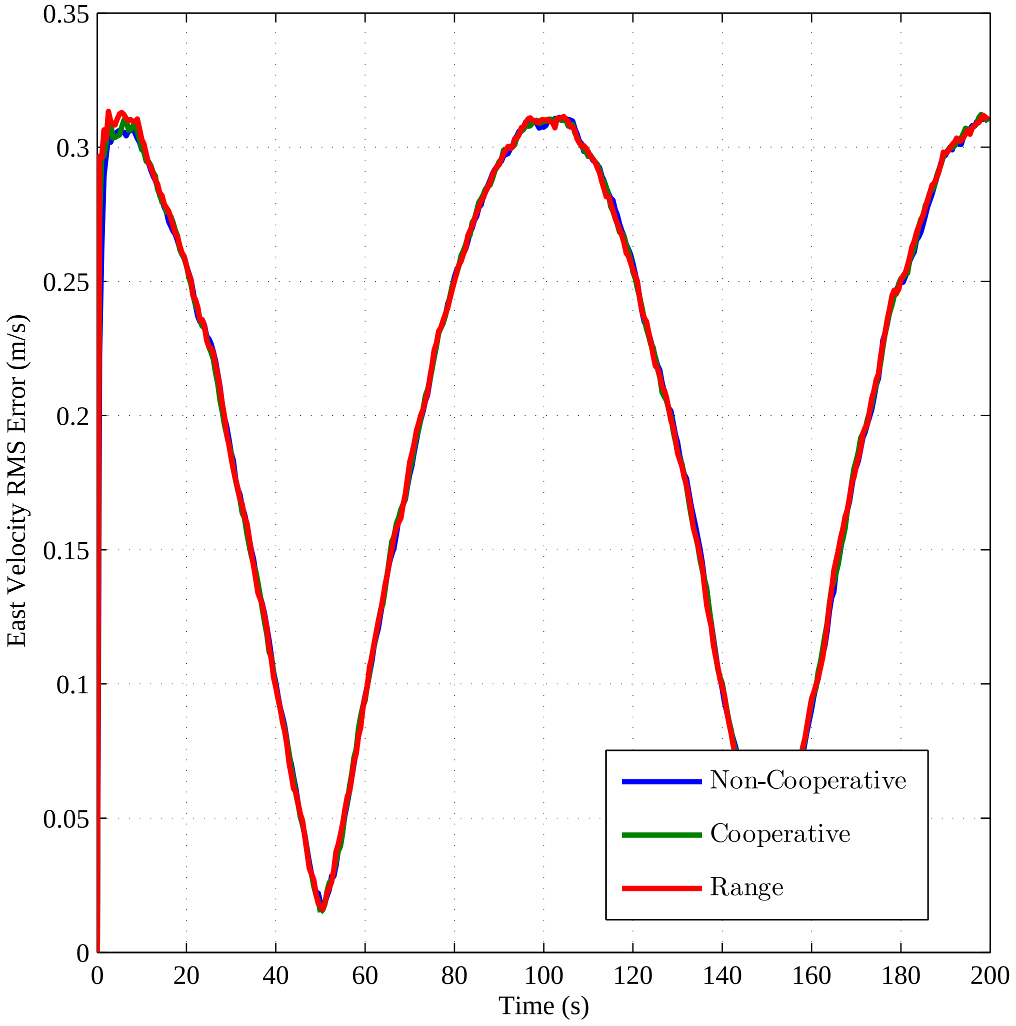


Figure 4.51: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the east axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

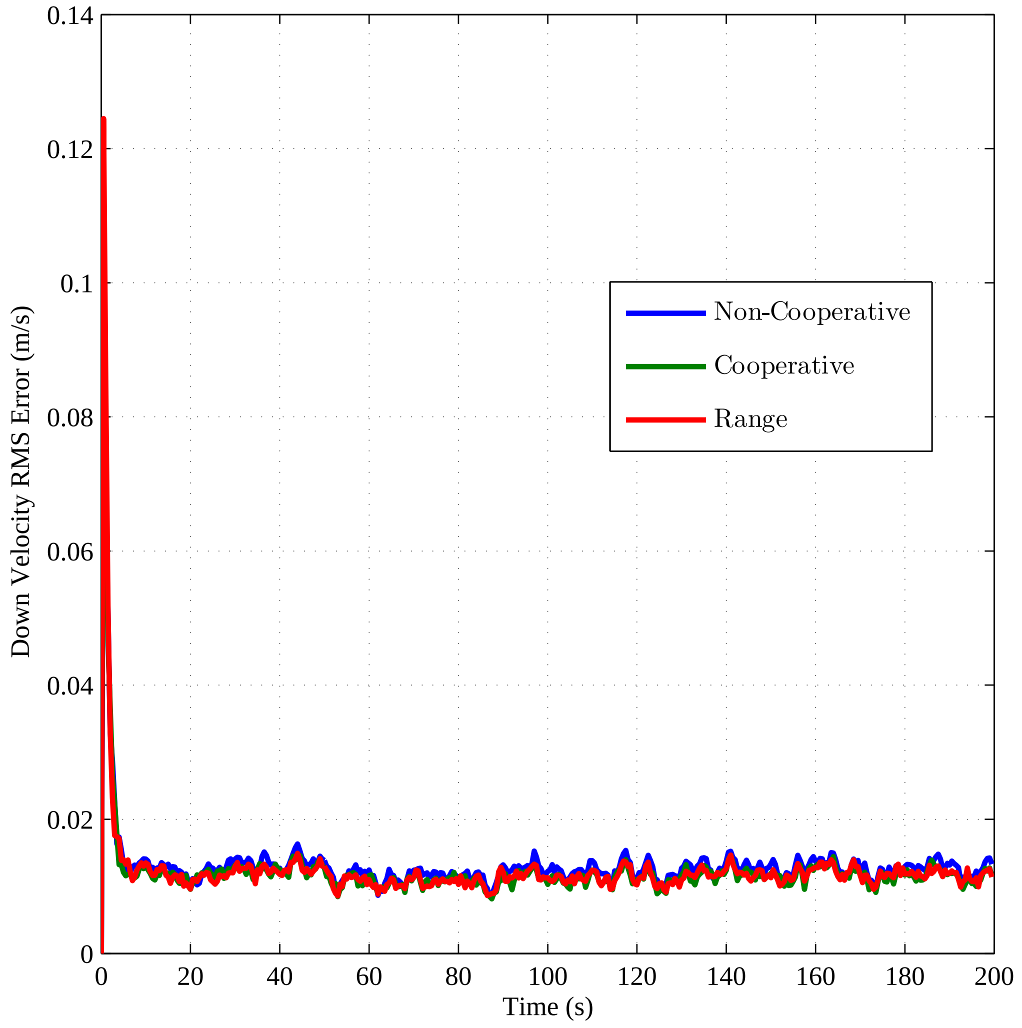


Figure 4.52: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

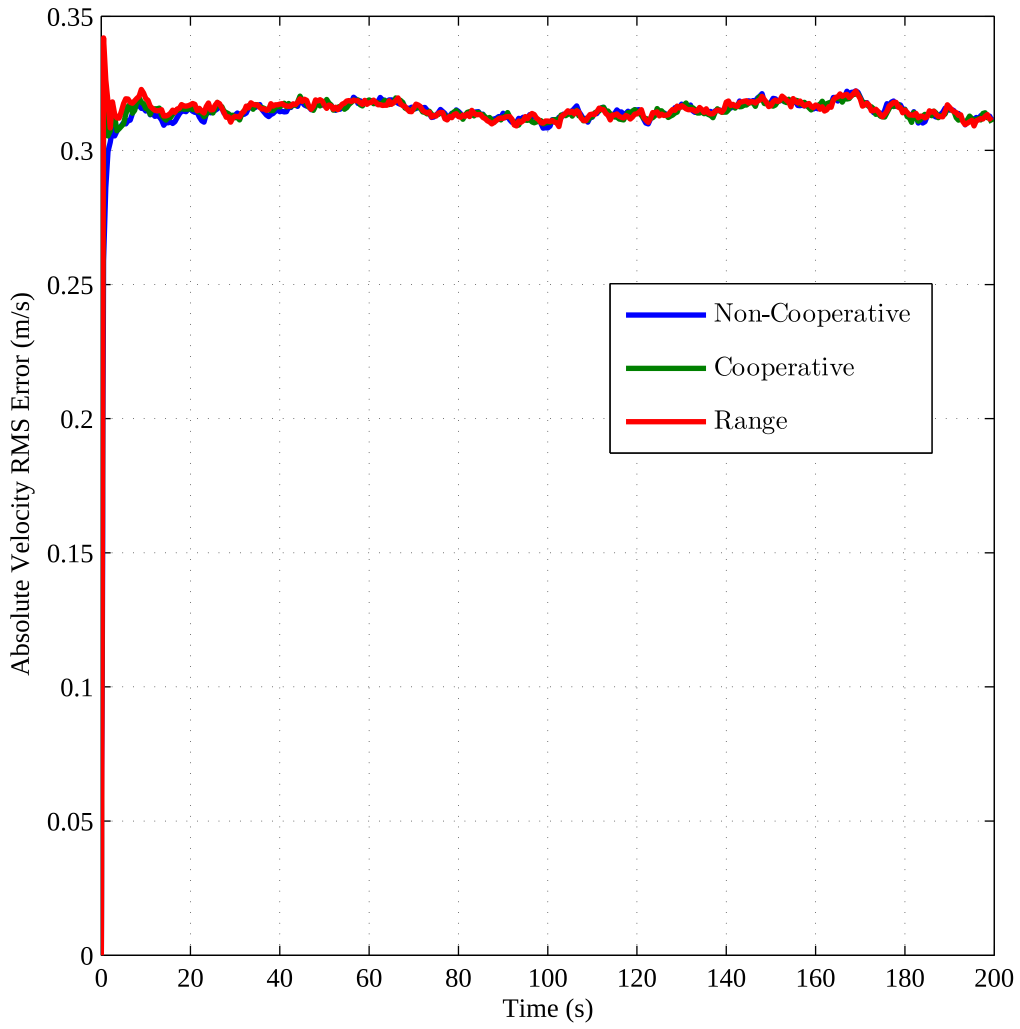


Figure 4.53: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute velocity error results . The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

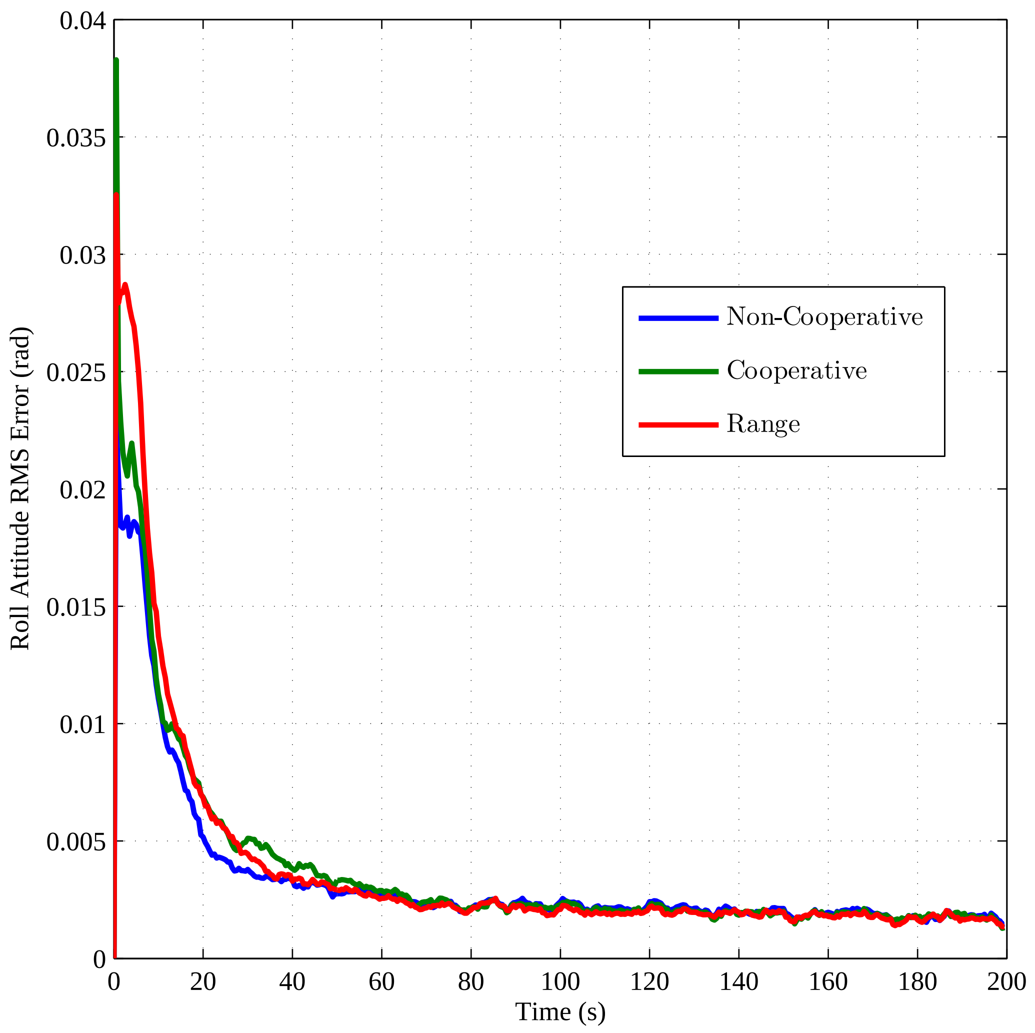


Figure 4.54: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

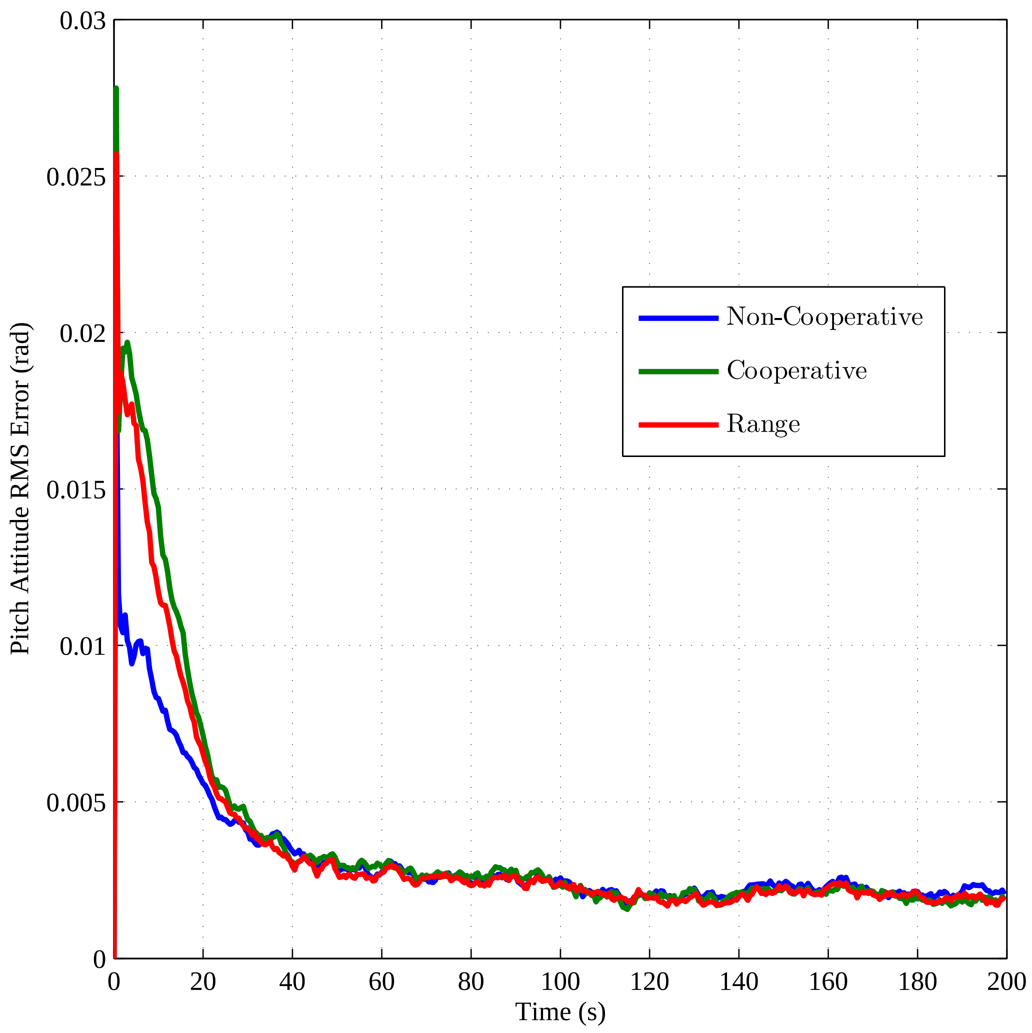


Figure 4.55: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

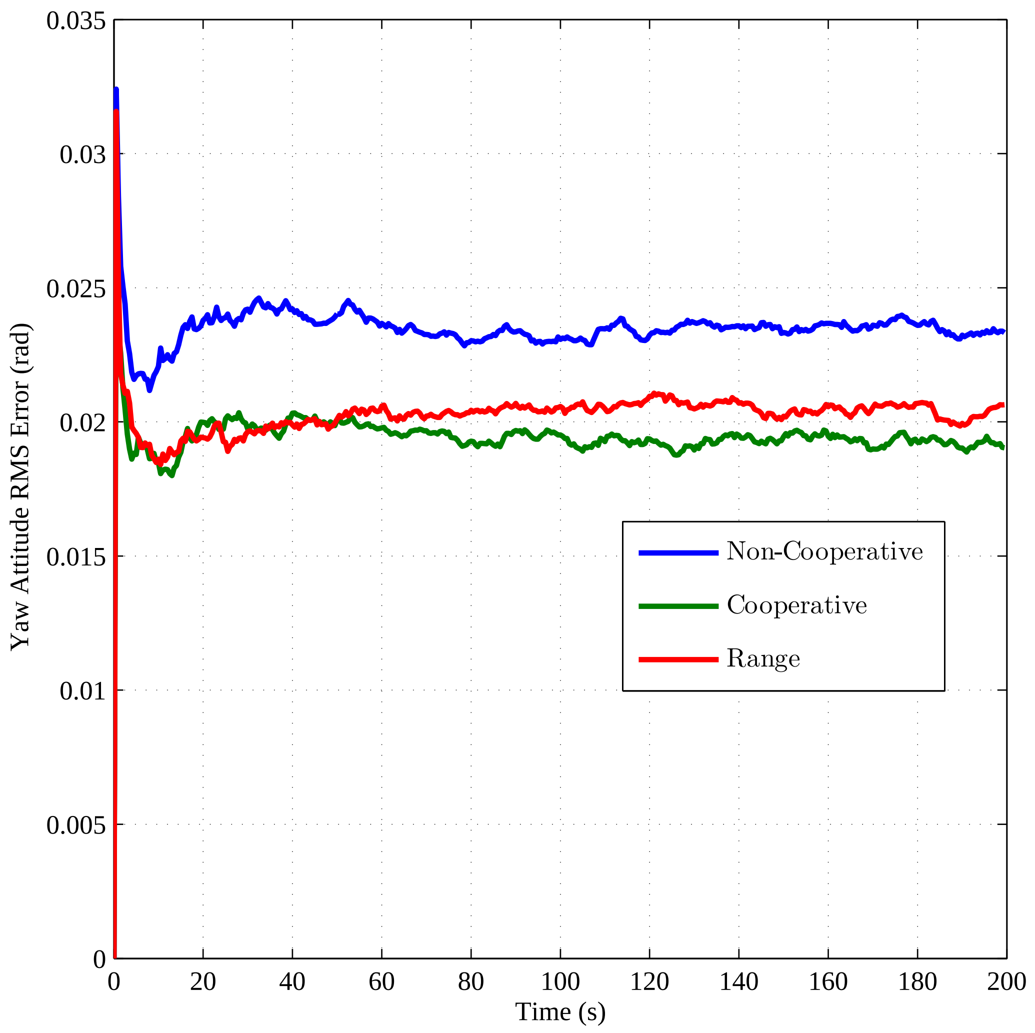


Figure 4.56: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

4.3.4 Yaw. The final pre-scaling simulation is designed to test the effect of an attitude maneuver, without any position change, to provide observability of the attitude error states.

The simulation is composed of two platforms positioned $1\,m$ above and below the *n-frame* origin respectively. The platforms begin the simulation pointed north and rotate around the down axis counterclockwise, completing one complete rotation over the $200\,s$ simulation. One hundred fifty reference features are uniformly distributed around a cylinder, centered on the *n-frame* origin, with a height and radius of $10\,m$. Figure 4.57 depicts the simulation scenario.

Figures 4.58-4.68 show the results of this simulation. These results are used, with the prior simulation results, in the next section to explain the attitude errors in the stationary and hallway simulations.

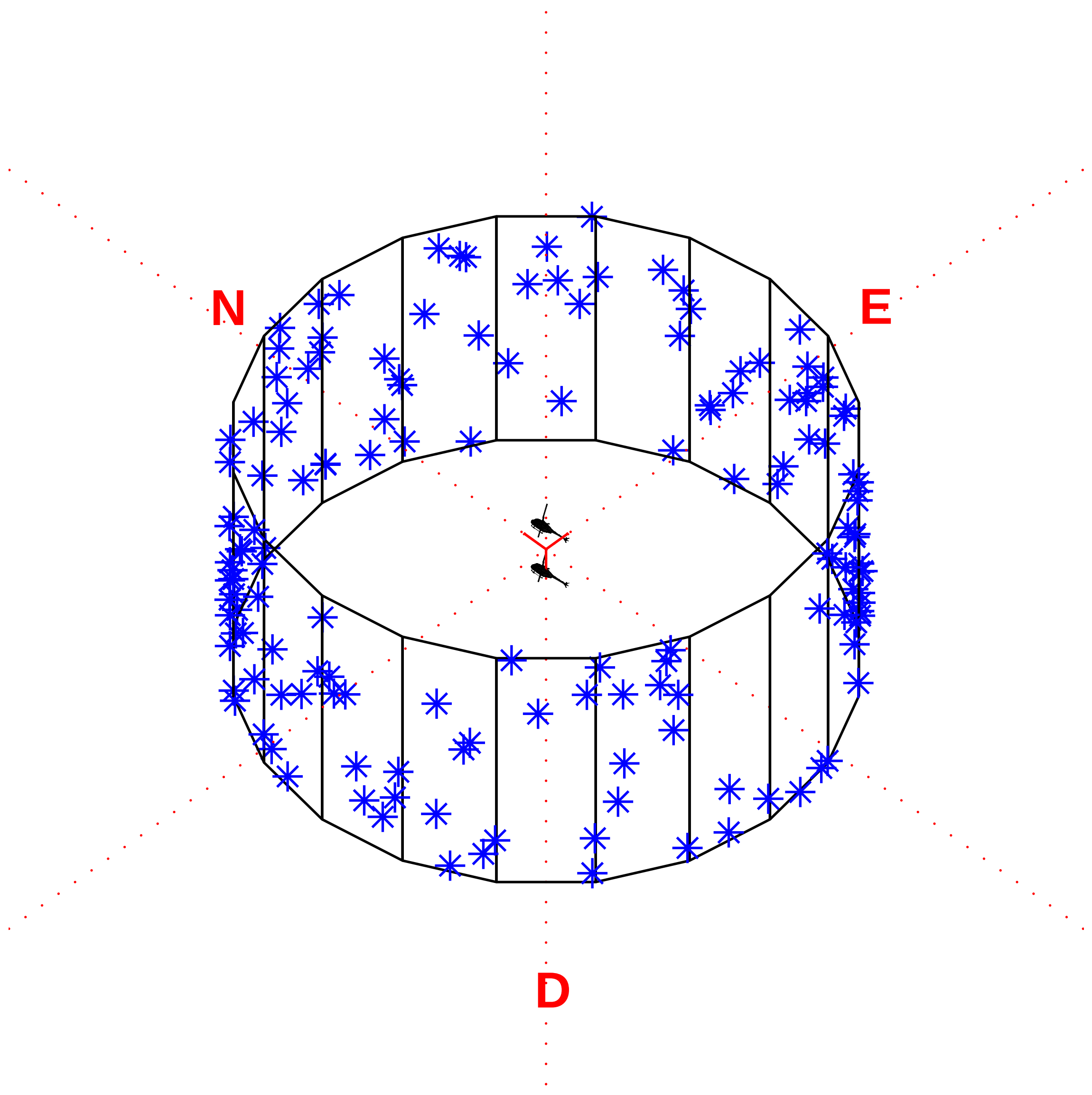


Figure 4.57: Two platform rotation simulation scenario using the pre-scaling method. The platforms each perform a counterclockwise 360° yaw motion over the course of the 200 s simulation. The two platforms begin and end the simulation heading north, positioned 1 m above and below the local navigation frame origin. One hundred fifty reference features are uniformly distributed around a cylinder, centered on the origin, with a height and radius of 10 m.

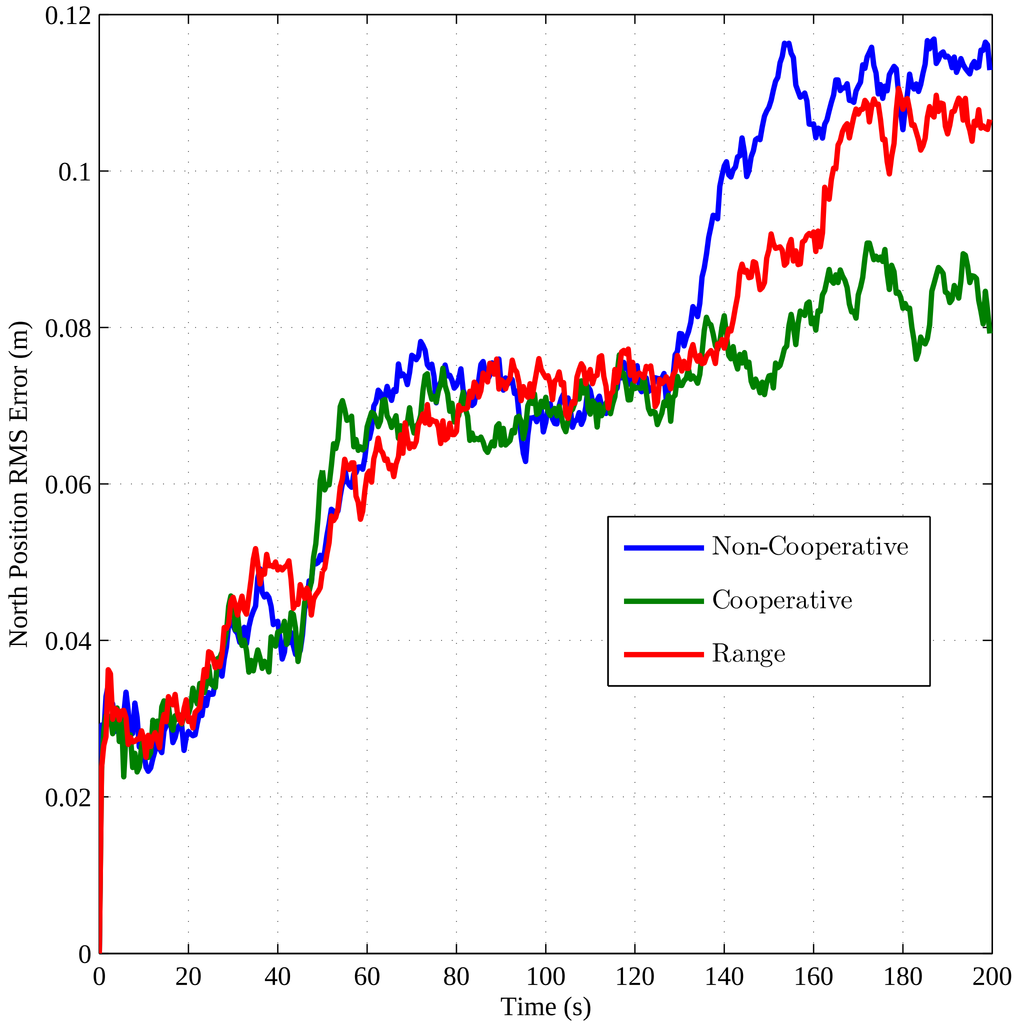


Figure 4.58: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

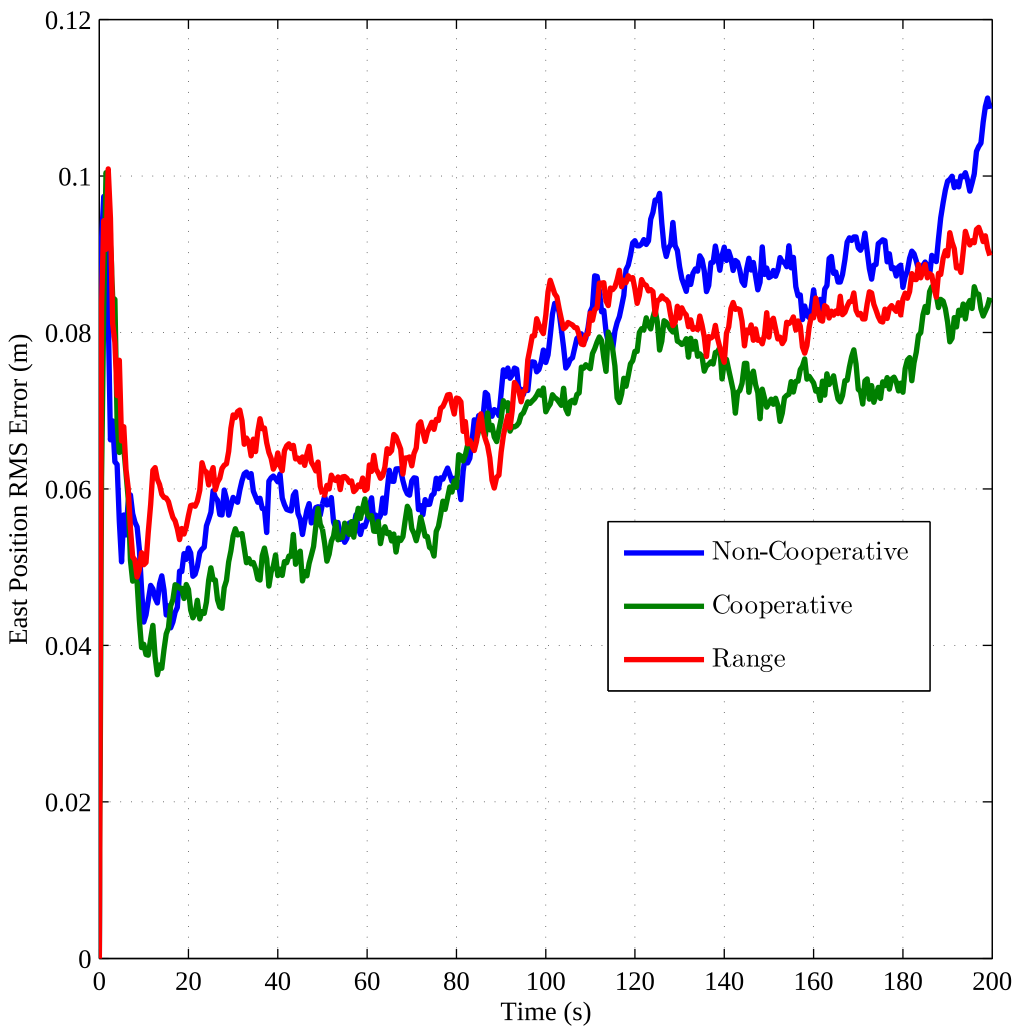


Figure 4.59: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the east axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

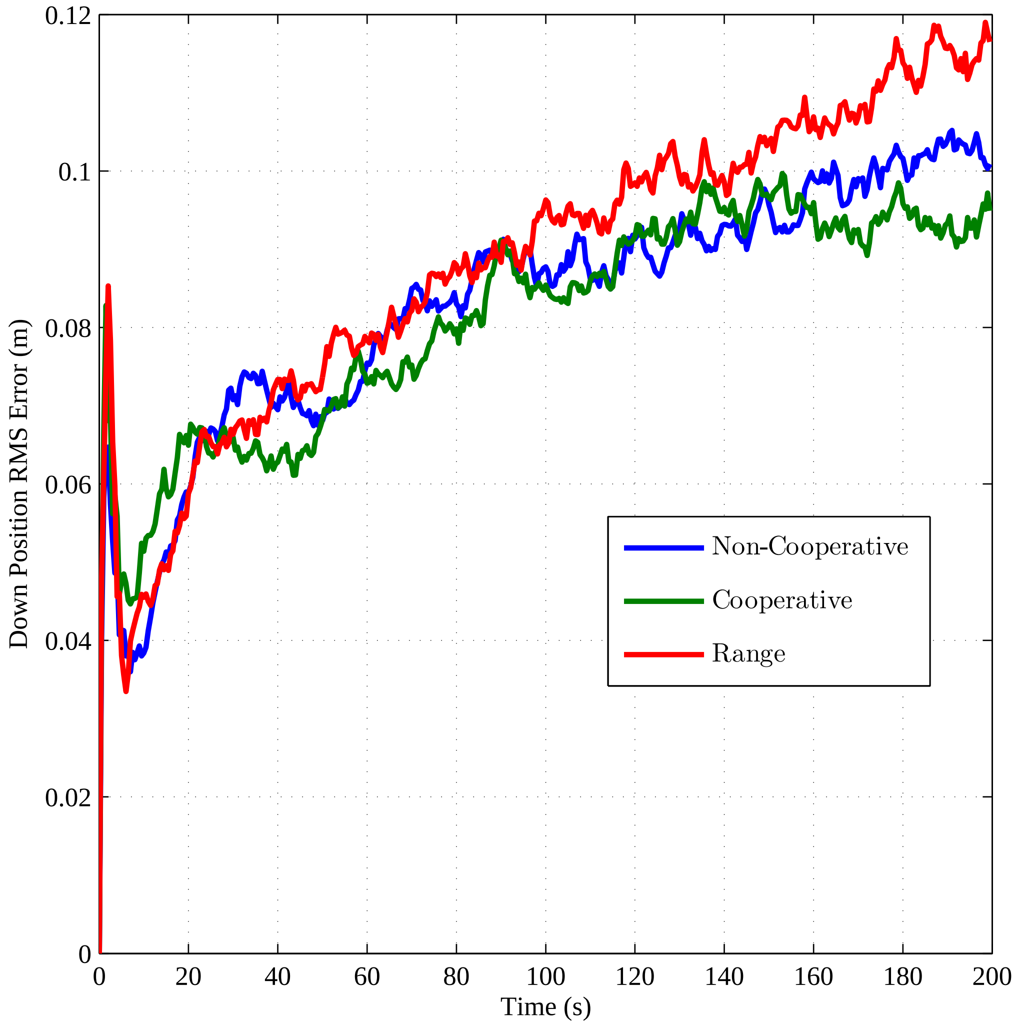


Figure 4.60: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

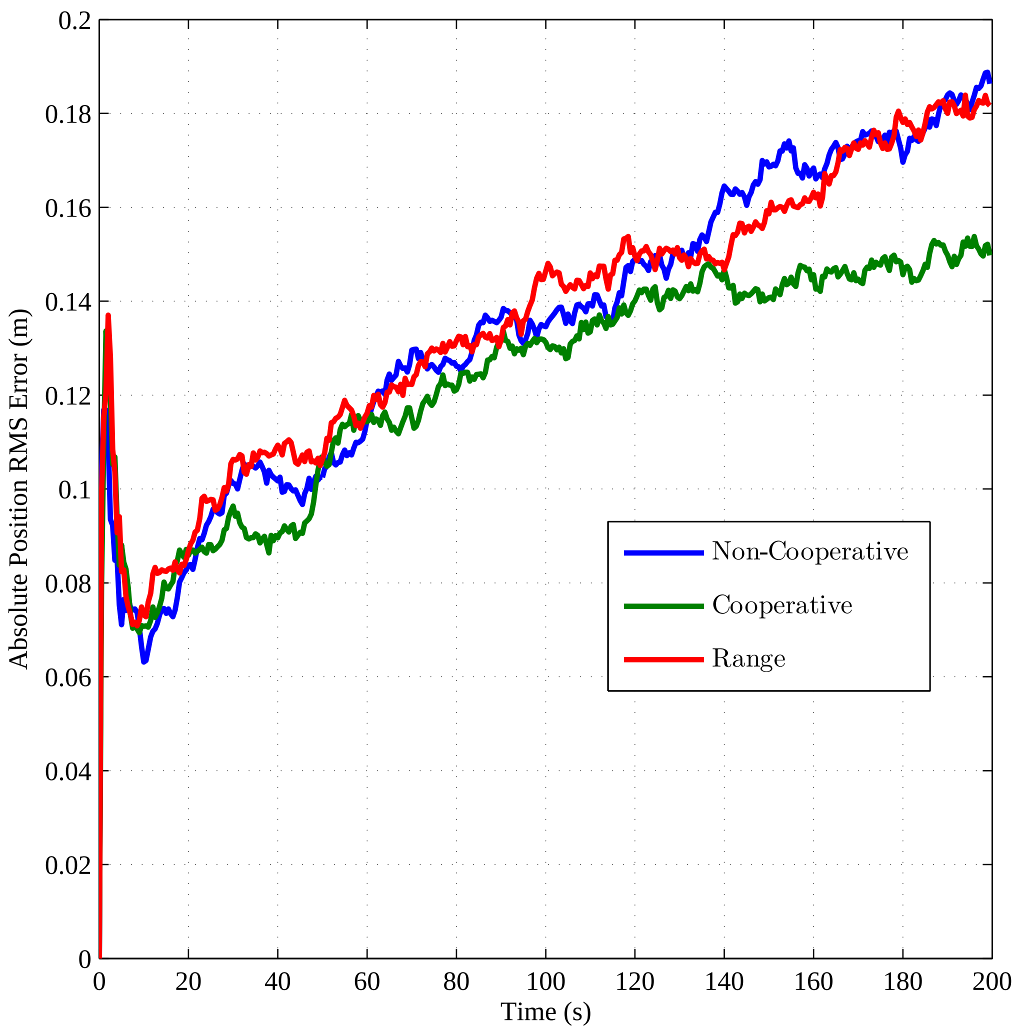


Figure 4.61: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute position error results. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

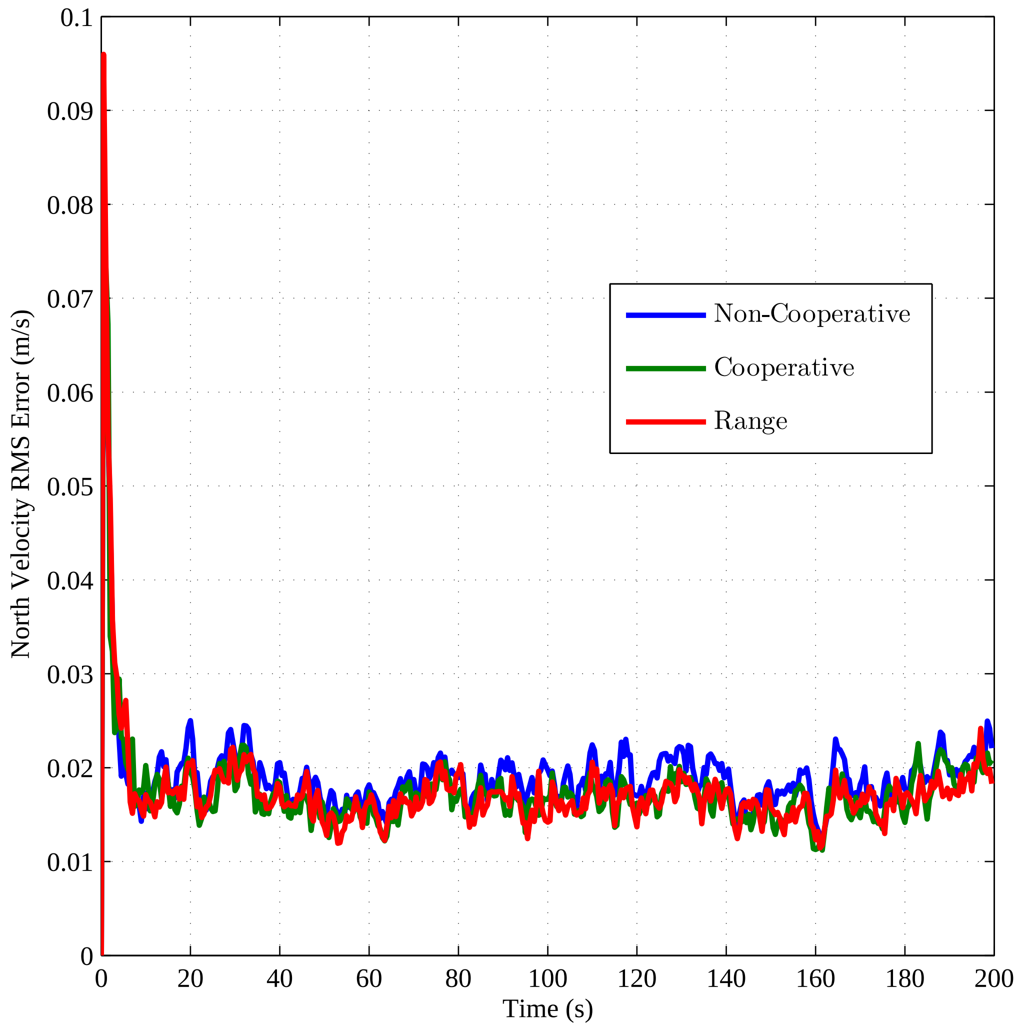


Figure 4.62: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a 10 m radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

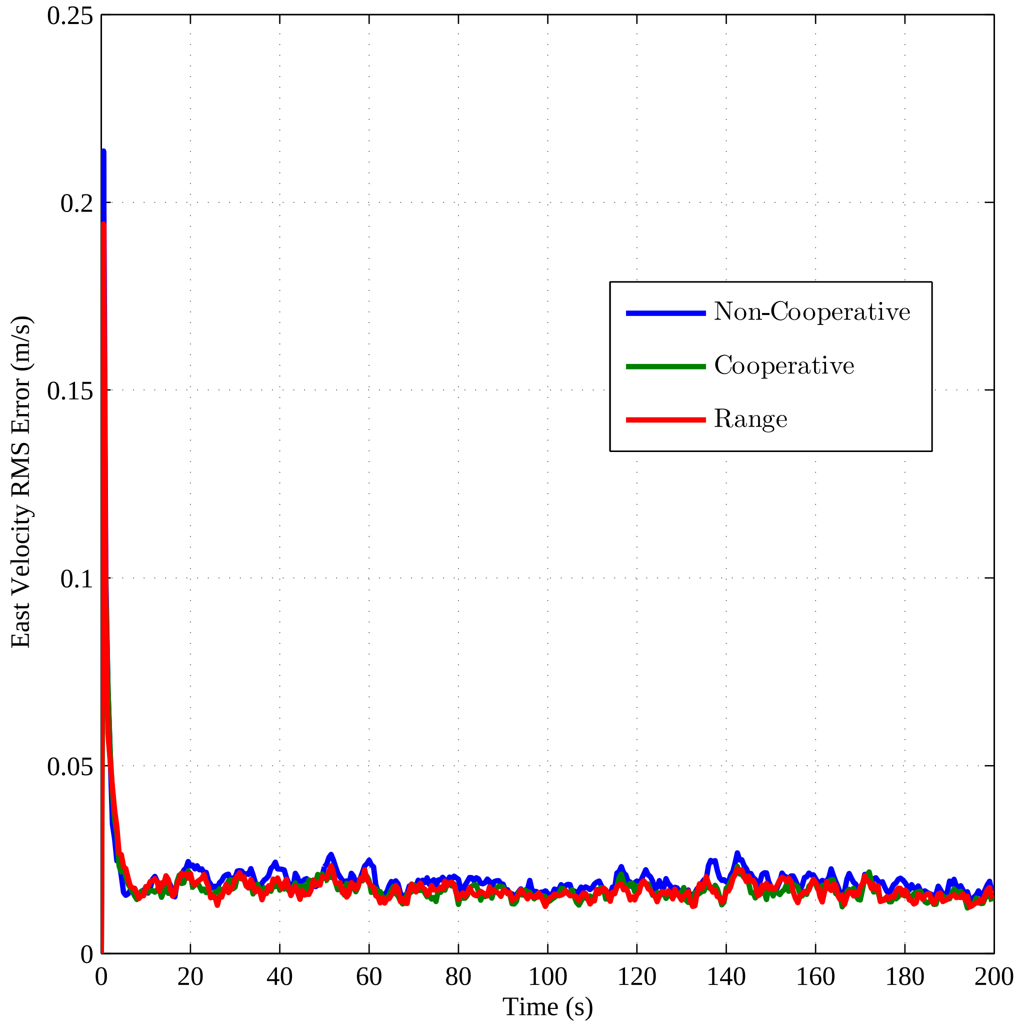


Figure 4.63: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the east axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

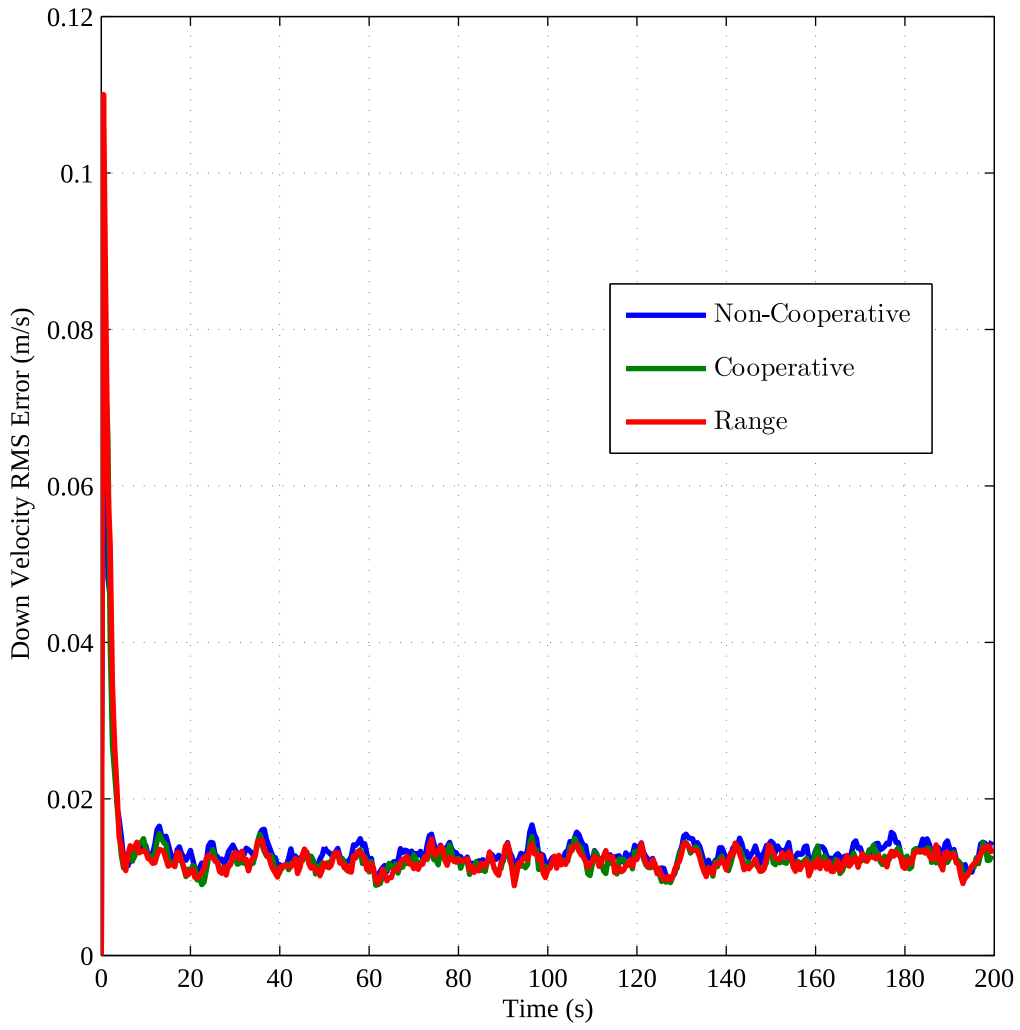


Figure 4.64: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a 10 m radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

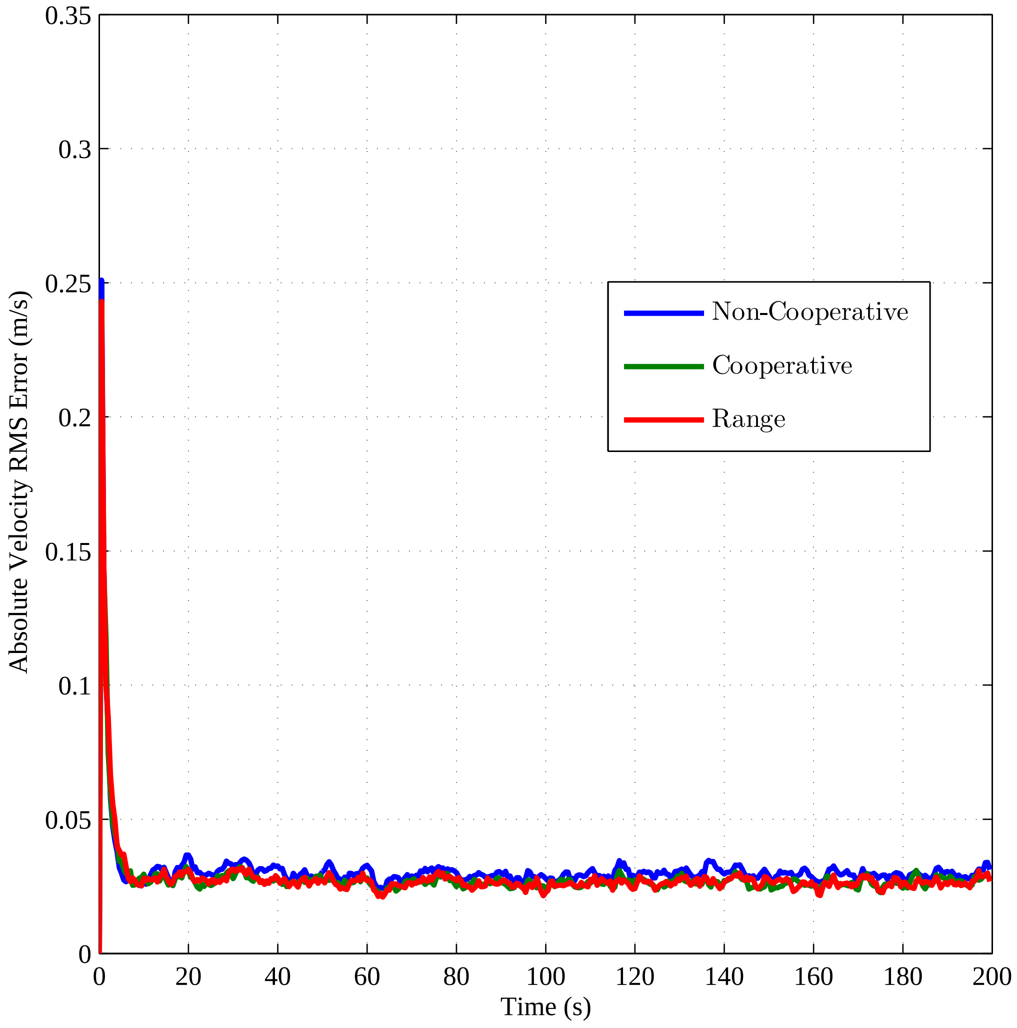


Figure 4.65: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute velocity error results . The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

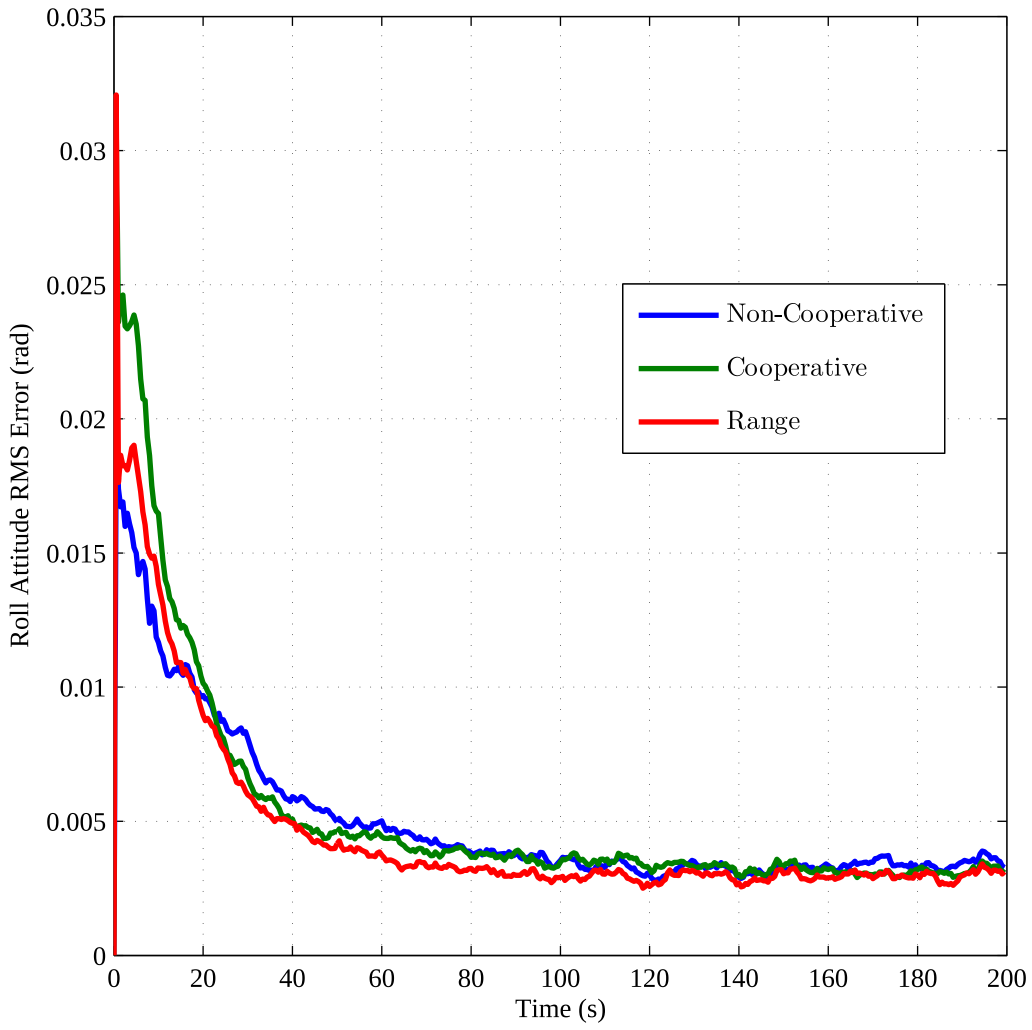


Figure 4.66: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

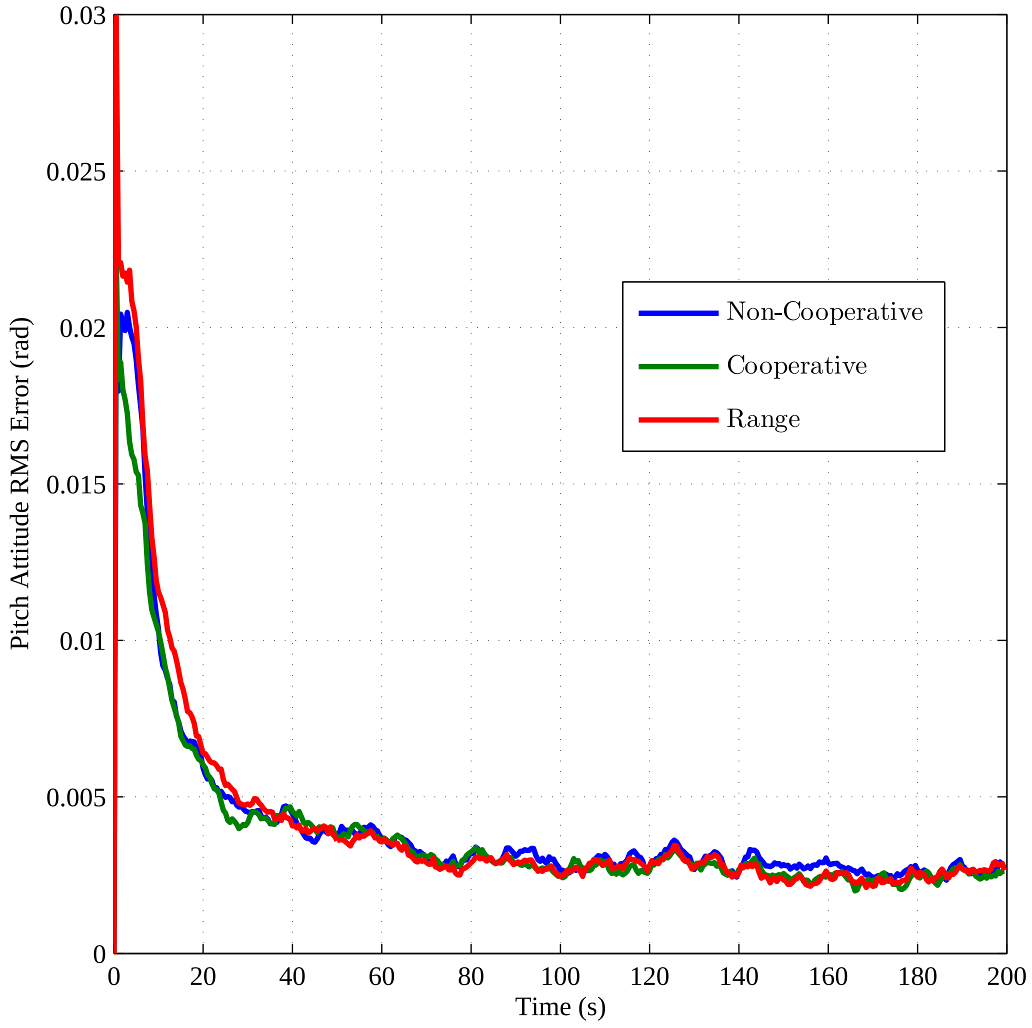


Figure 4.67: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

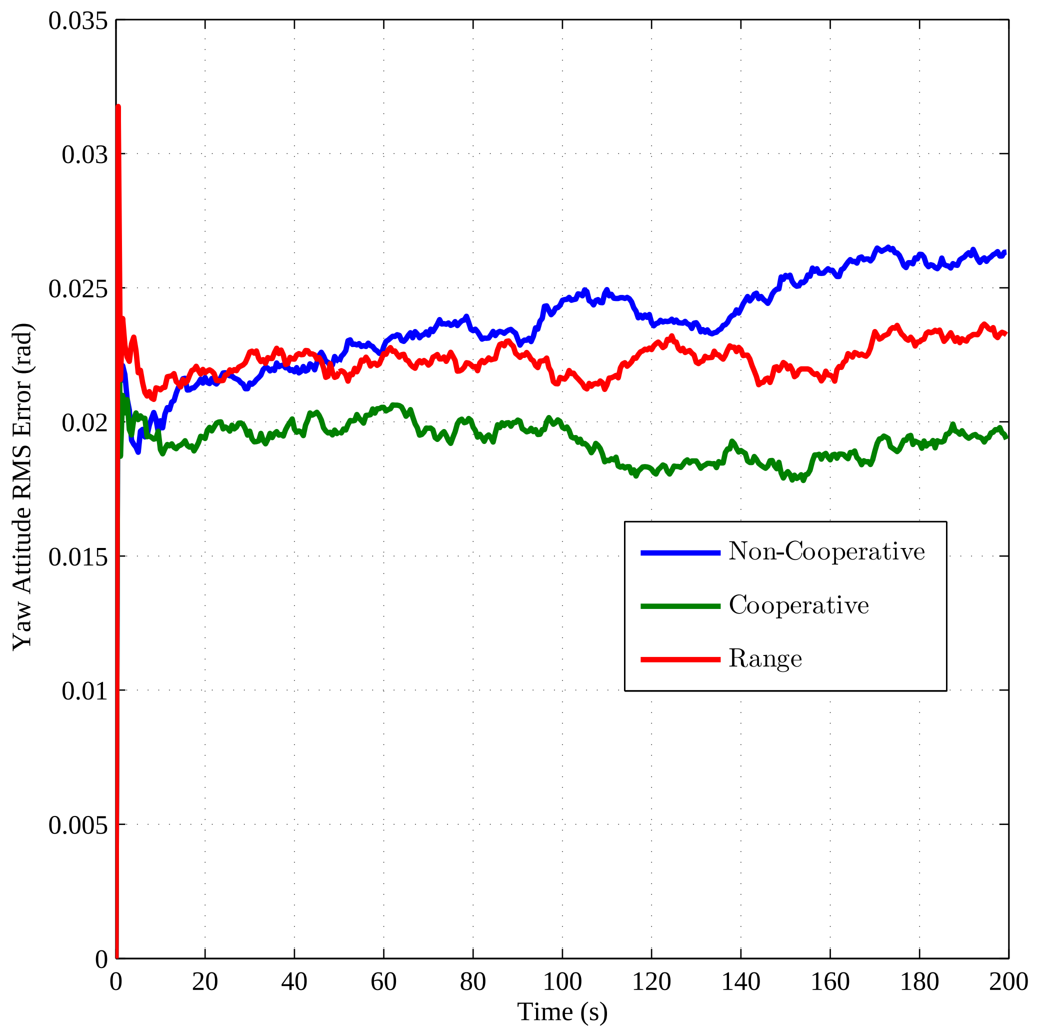


Figure 4.68: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

4.3.5 Attitude Errors. The results of the four simulations show two factors combining to cause the increased attitude errors seen in Figures 4.30-4.32 and 4.42-4.44. These two factors are decreased observability of the platform's attitude in the landmark positions, and the scaling of local filters in federated filter.

Figures 4.69-4.71 show the non-cooperative system results from the four simulations performed to verify the operation of the cooperative systems. These results show a yaw maneuver redicing the pitch and roll attitude errors of the non-cooperative system. They demonstrate a rotational maneuver improving the attitude error in the two opposite directions.

These results show attitude maneuvers providing observability of the attitude in the landmark information. Therefore, in the stationary and hallway simulations, where no attitude maneuvers are performed, the shared landmark information contains very little attitude information.

The scaling of the local filter information in a federated filter provides the second factor which increases the attitude errors. Combining the reduced attitude observability with the local filter scaling mechanism creates the increased attitude errors in the stationary and hallway simulations. In both simulations the information from the primary filter is halved and is not compensated by the secondary filter as the landmark information from the second platform contains little or no observable attitude information.

Theses results show how the execution of a yaw maneuver, in the sideslip and yaw simulations, provides increased attitude information in the shared landmark information and compensates for the scaling of the primary filter information. However, these results may be the result of the simulations scenarios and require further investigation to determine the root cause for the attitude errors.

In the next chapter, conclusions are drawn regarding the performance of the cooperative systems and recommendations are presented for future implementation of this research.

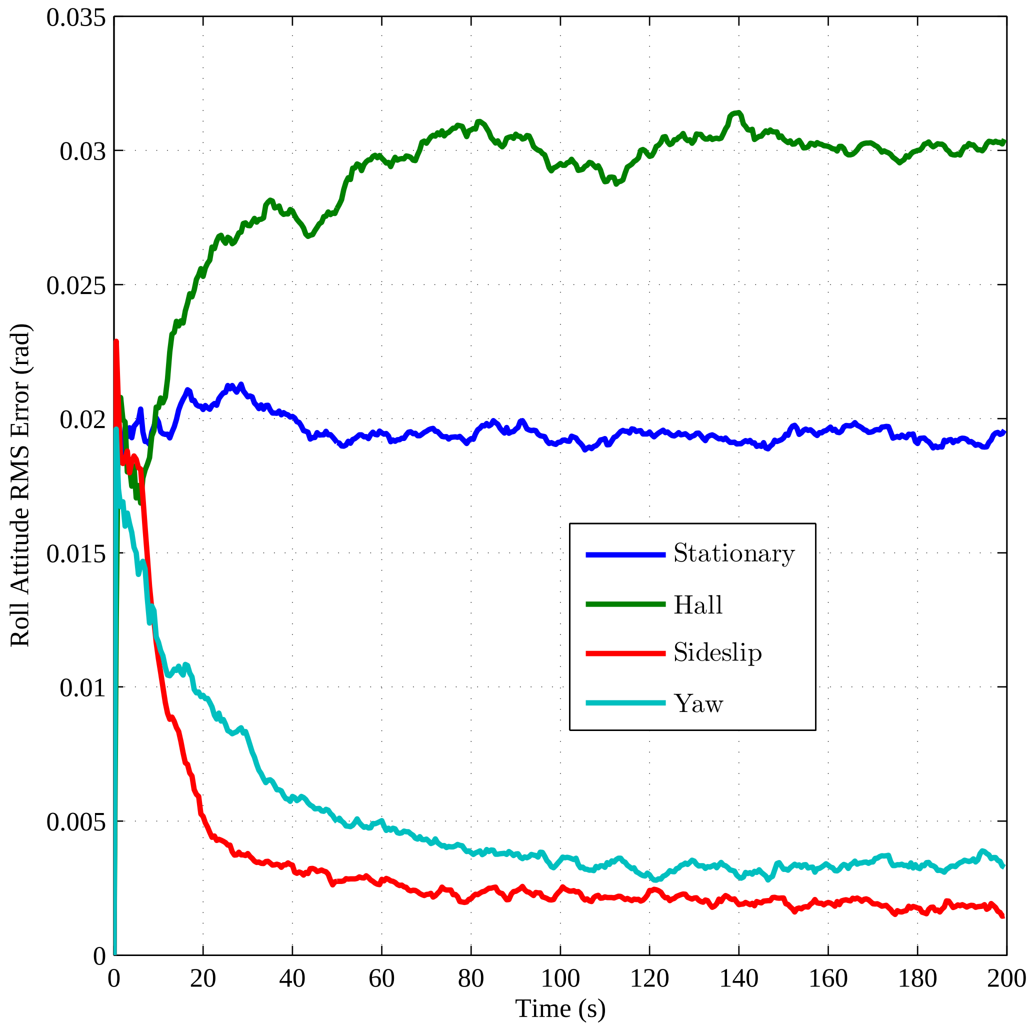


Figure 4.69: Non-cooperative system root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the stationary (blue), hallway (green), sideslip (red), and yaw (cyan) simulations.

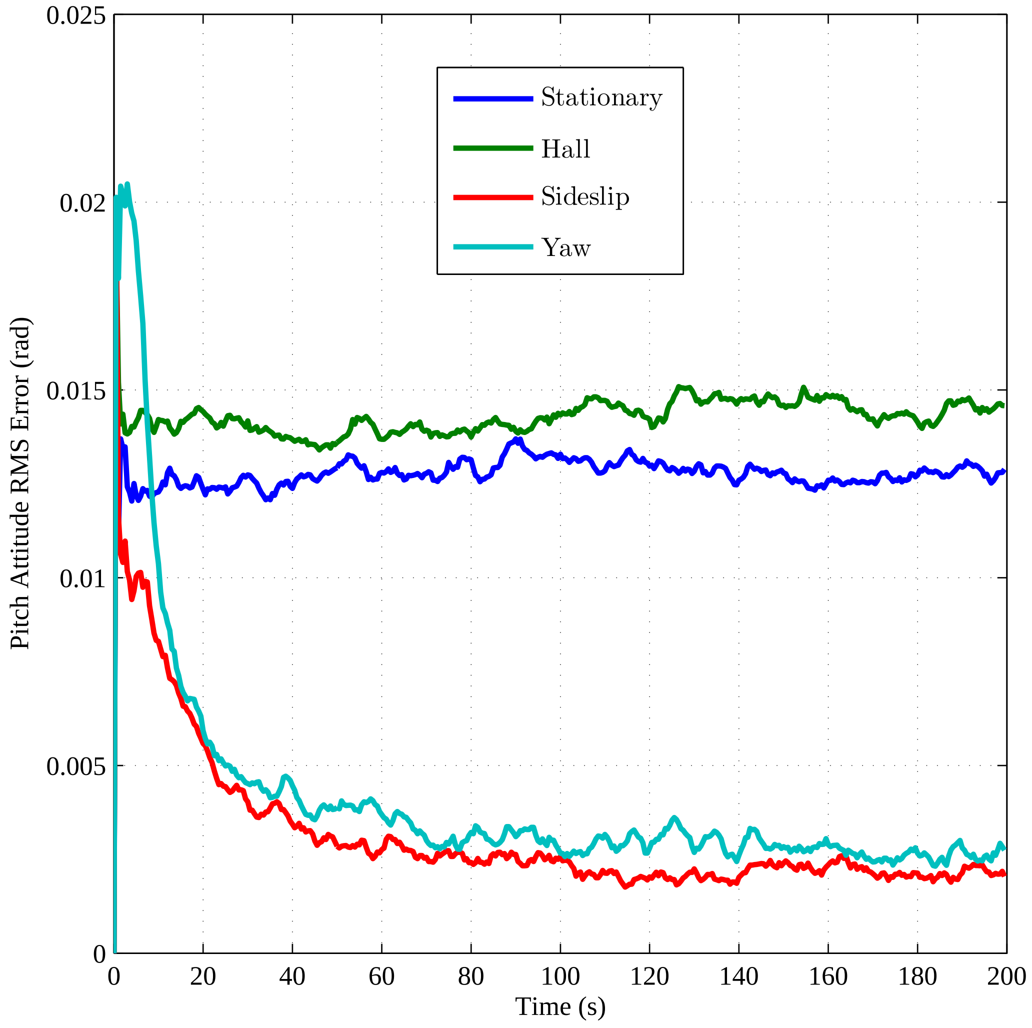


Figure 4.70: Non-cooperative system root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the stationary (blue), hallway (green), sideslip (red), and yaw (cyan) simulations.

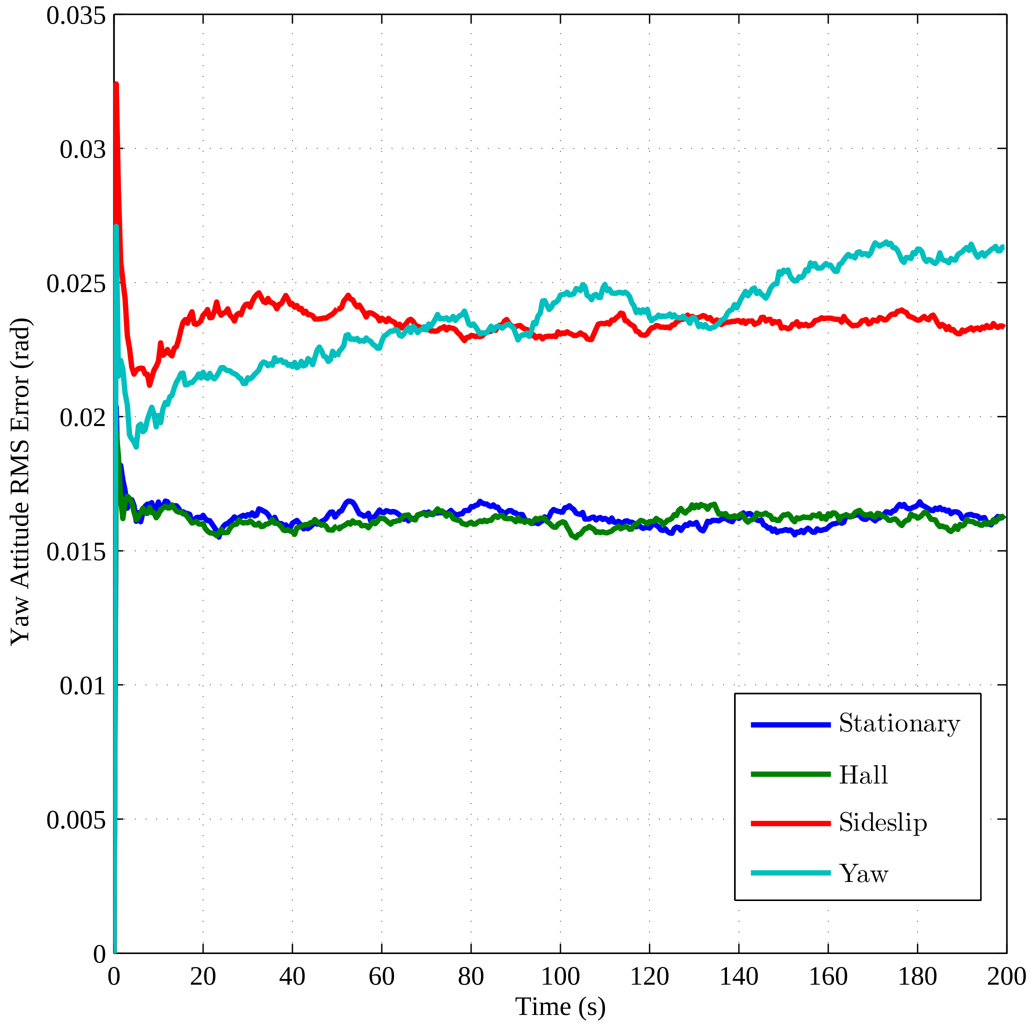


Figure 4.71: Non-cooperative system root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the stationary (blue), hallway (green), sideslip (red), and yaw (cyan) simulations.

V. Conclusions and Recommendations

This thesis details a research effort focused on expanding previous research into the fusion of optical and inertial sensors for robust, autonomous navigation to multiple platforms. In this chapter, conclusions regarding the research are presented and discussed along with recommendations for potential future research.

5.1 Conclusions

The most significant conclusion for this research is the demonstration of the viability of cooperative, vision-aided navigation using federated filtering techniques. The goal of this research is the development of a method for sharing landmark information, acquired through vision systems, between platforms improving the navigation solutions. The concept is developed into two possible cooperative navigation systems and tested using `Matlab`® simulation.

Both navigation systems utilize a federated filter to incorporate landmark information, and possibly platform ranging information, from secondary platforms into a fused total solution. The first system is designed to directly leverage the algorithms for a single platform vision-aided navigation system without modifications. This system requires the implementation of a new federated filter scaling system which allows for the scaling of the local filters after the propagation and measurement update steps are performed. The second system is designed to use the single platform algorithms after modification to allow for a direct implementation of a federated filter.

Simulation of the first navigation system shows no change in the accuracy of the cooperative system compared to the single platform or non-cooperative system. The cooperative system is unable to improve on the non-cooperative results due to the method by which the primary filter is implemented. The primary filter implements the single platform algorithms without modification. Therefore, the primary filter solution is computed at every time step before the solution is applied to the master filter. This prohibits the inclusion of external data into the solution to improve navigation state accuracy. Further investigation is required to determine if the post-scaling method developed in Chapter III is valid.

As expected, simulation of the second cooperative navigation system demonstrates an ability to reduce the position and velocity errors over a non-cooperative vision-aided system. However, when no rotational motion is present in the overall motion of the platforms, the cooperative systems increase the attitude errors as demonstrated in Figures 4.30-4.32. This increase of the attitude errors is due to the scaling of the local filters and reduced observability of the attitude information in the absence of rotational motion. Additionally, when platform motion is limited to linear position changes the attitude errors couple the position errors in the direction of motion to the off axis directions. This phenomenon is demonstrated in Figures 4.34-4.40.

There exist two methods to reduce or overcome the attitude errors caused by a lack of rotational motion: *a*) adaptive scaling of the local filters, and *b*) inducing rotational motion.

Adaptive scaling of the local filters require using separate scaling parameters for the platform and landmark portions of the covariance matrix. The scaling parameter for the platform portion is set to a constant while the scaling parameter for the landmark portion is adaptively varied dependent on the amount of rotational motion in the secondary platform. The adaptive scaling method will reduce the amount of information provided by the secondary platform and increase position errors while reducing the attitude errors. Additionally, this method is difficult to implement.

The introduction of rotational motion to the cooperating platforms is a much easier solution to implement. In fact, many real world systems may naturally include enough rotational motion to counteract the errors shown in the performed simulations. However, these errors should not be overlooked when designing control systems which are provided navigation solutions using this cooperative navigation system.

While the approach developed in this research demonstrates improved performance, over non-cooperative navigation systems, there are areas which merit additional research. These areas are addressed in the next section.

5.2 Recommendations

This section recommends and briefly discusses three areas of possible interest for future research.

The first area of future research is to investigate the effect of communication protocols and networking techniques upon the cooperative navigation system. The research presented in this thesis assumes a common communication protocol and does not implement any form of networking. Additionally, the simulations performed in Chapter IV implements the communication between platforms instantaneously. Any efforts to implement the cooperative navigation system concept on real-world systems must account for the latency in communications and network topology.

The next area of future research is the further development and implementation of the concepts developed in this research to real-world systems. This research would first entail performing new simulations using data collected from real-world systems to validate the cooperative navigation system concepts with data from real sensors. Next, the algorithms developed for this research require updates to allow for real-time implementation. Finally, implementation of the real-time algorithms onto unmanned vehicles will allow for the experimental validation of this research.

The final area of future research is too further investigate the observability of attitude information in the landmark data. As discussed in the previous section, the observability of attitude information in the landmarks is severely reduced when the platforms do not perform rotational maneuvers. This poses a problem in the cooperative navigation system when linear position errors are increased due to the coupling of errors between axes. As demonstrated in Section 4.3.2 increased attitude errors increased the coupling of position errors between axes and reduce the effectiveness of the cooperative navigation system. This research would entail investigating the attitude observability and determining the optimal means for mitigating the increased attitude errors.

5.3 Summary

This thesis presented the problem of expanding past research into the coupling of vision and inertial navigation systems to multiple cooperating platforms. A federated filter was designed to provide a solution and using `Matlab`® simulations the validity of the filter was explored.

The federated filter designed in this thesis is a first step in the development of a cooperative navigation system utilizing fused vision and inertial sensor data. The development of such a system has only just begun and, as discussed in the previous section, there is abundant work required to realize the full potential of this technology.

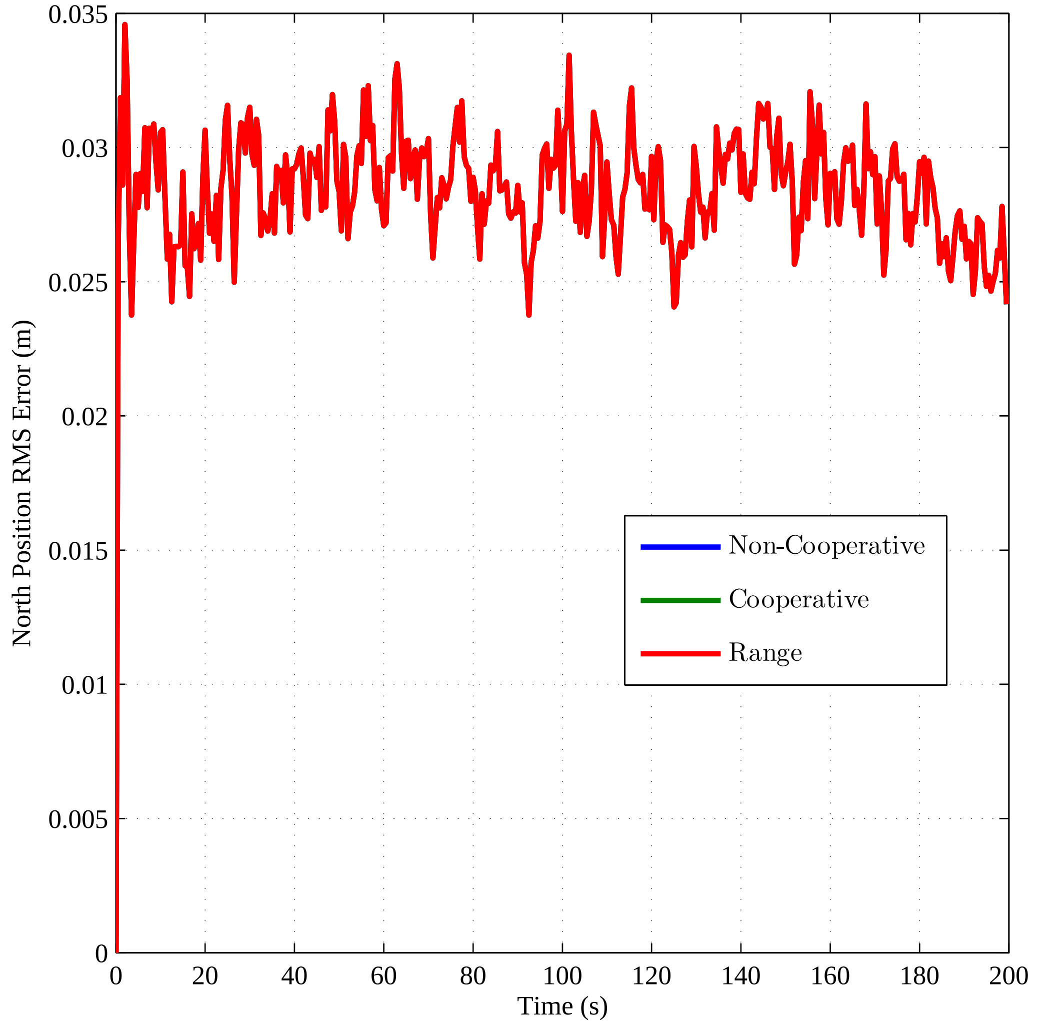


Figure A.1: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

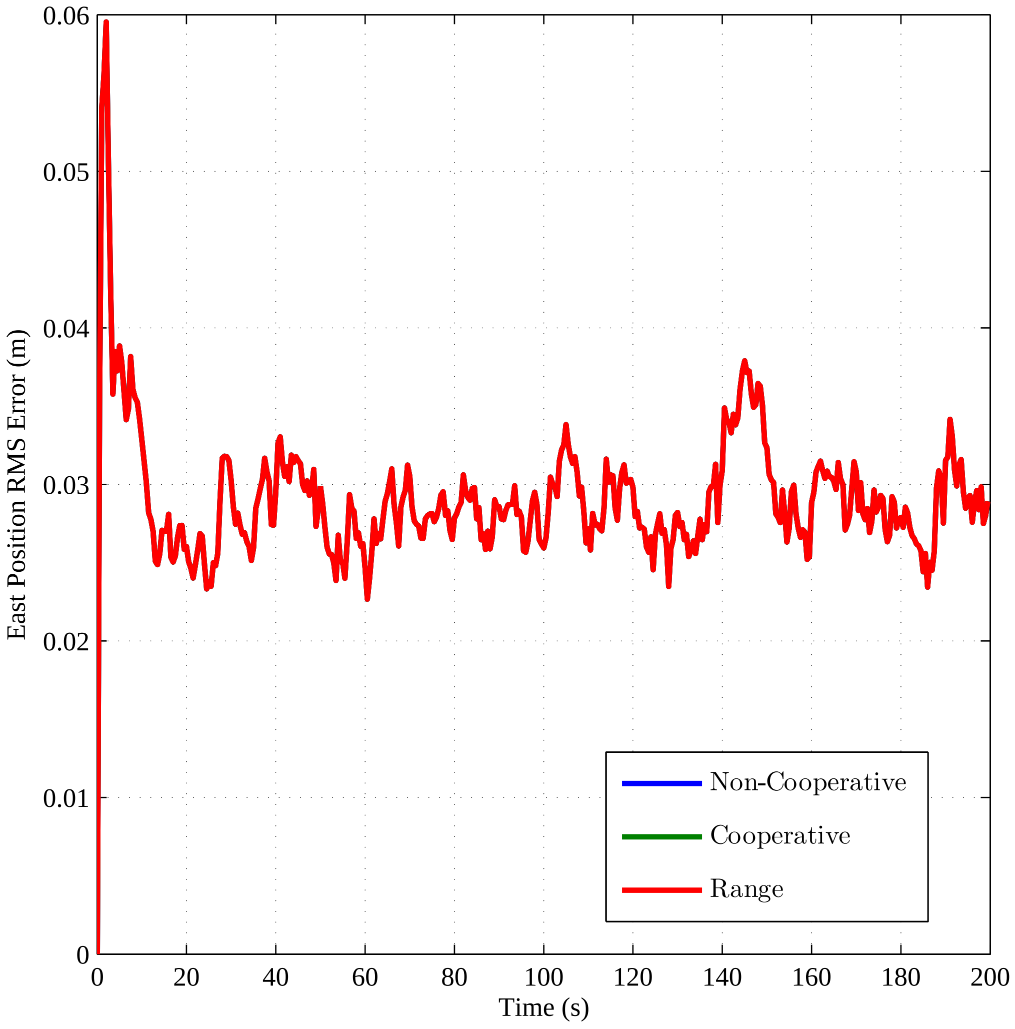


Figure A.2: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

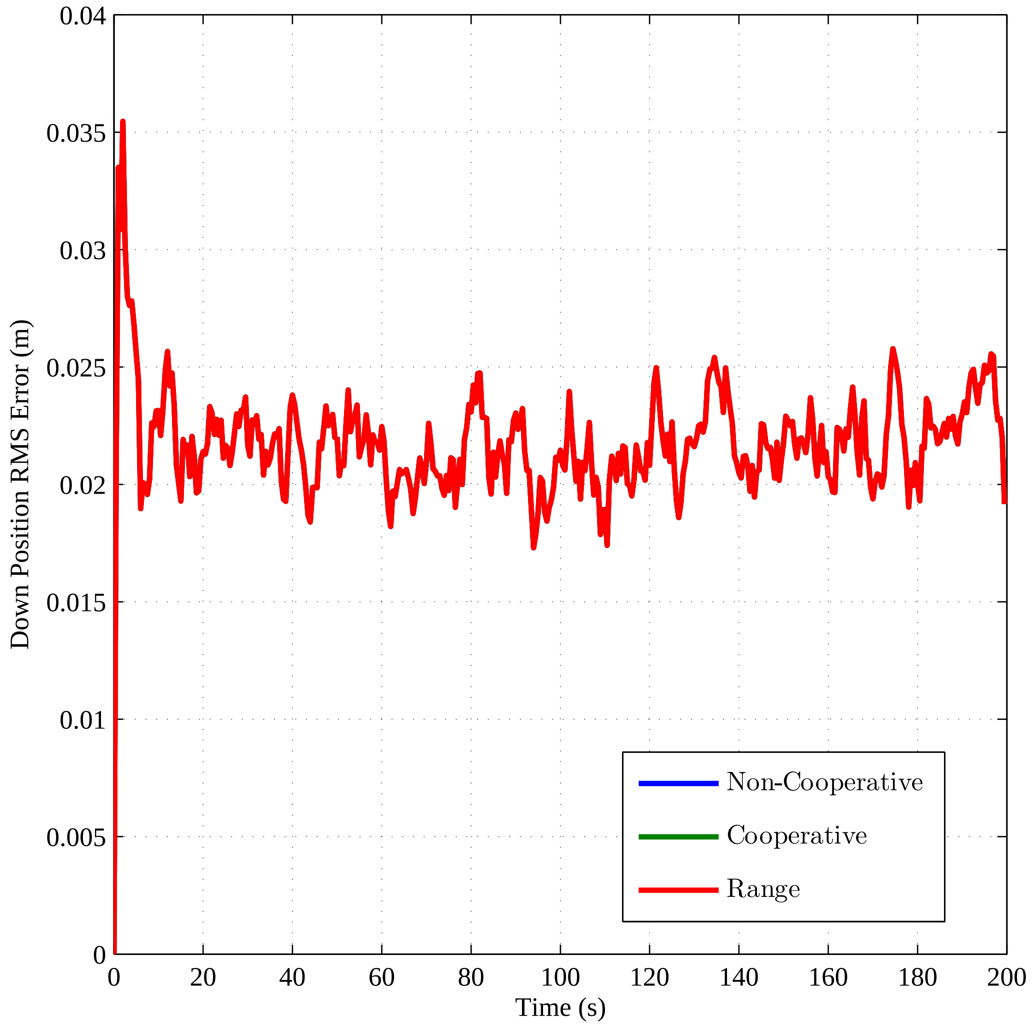


Figure A.3: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

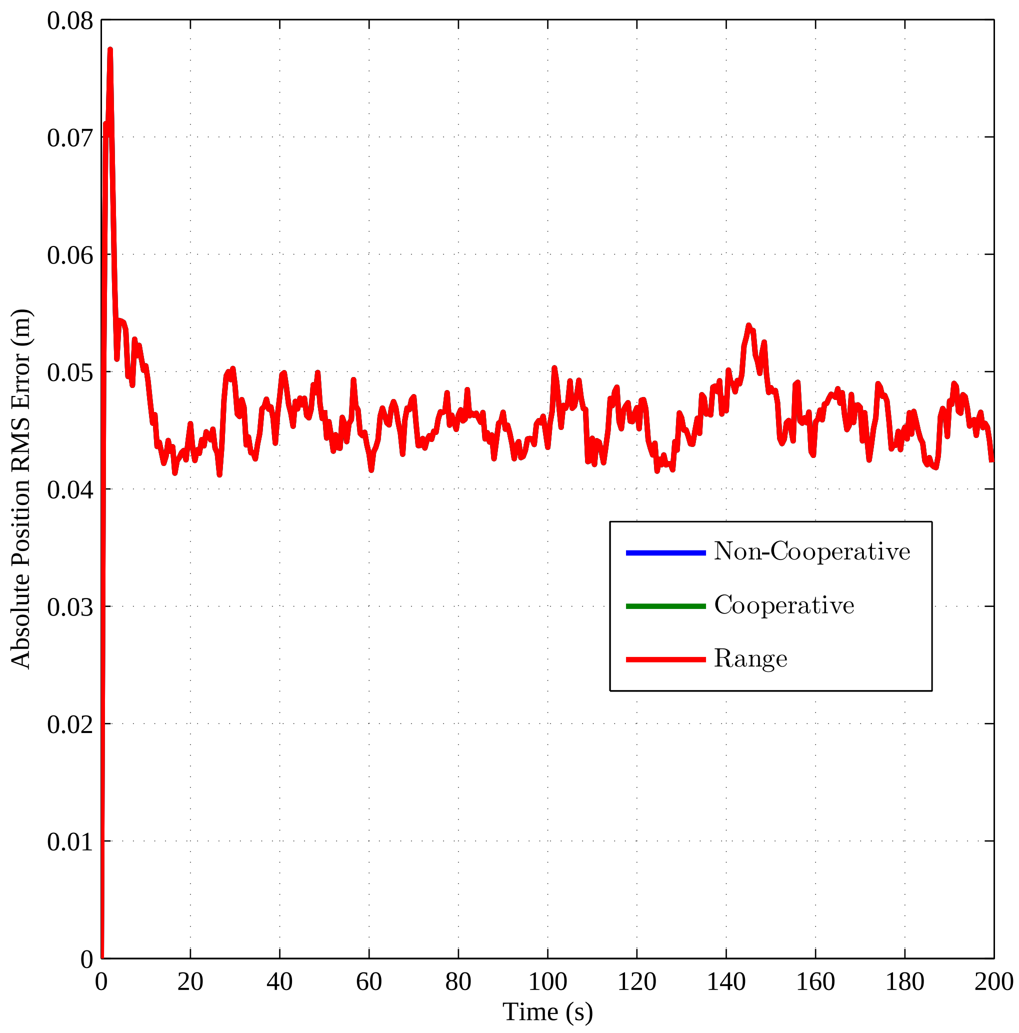


Figure A.4: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute position error results. The results shown are for the first of two stationary platforms simulated using three system implementations: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

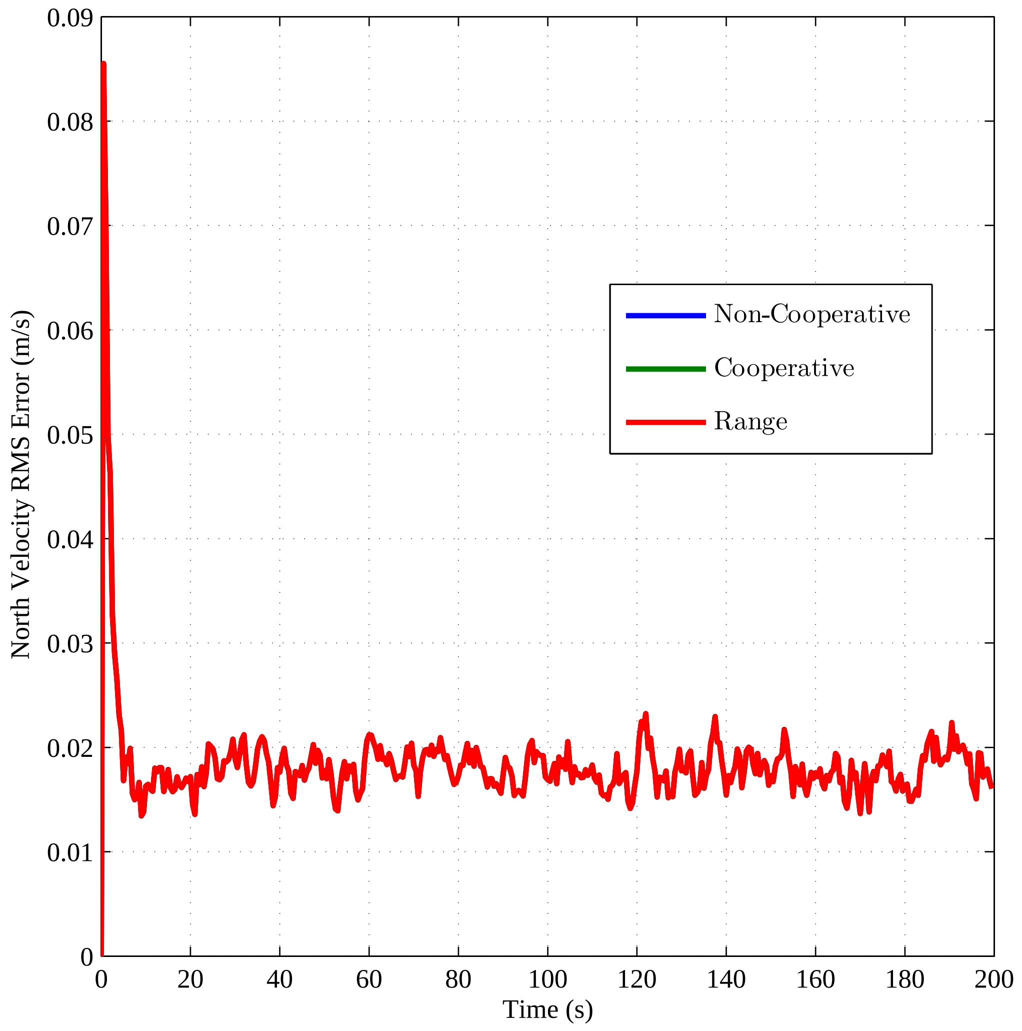


Figure A.5: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

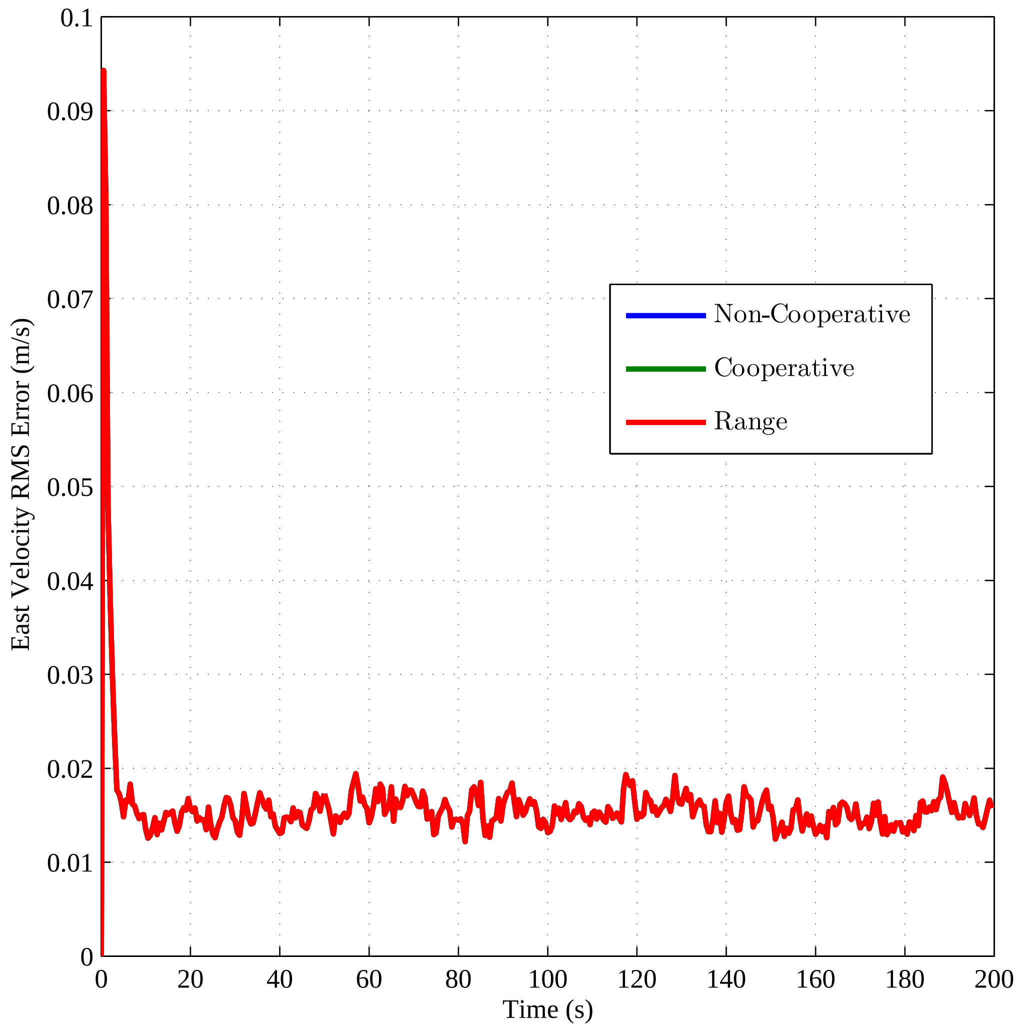


Figure A.6: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

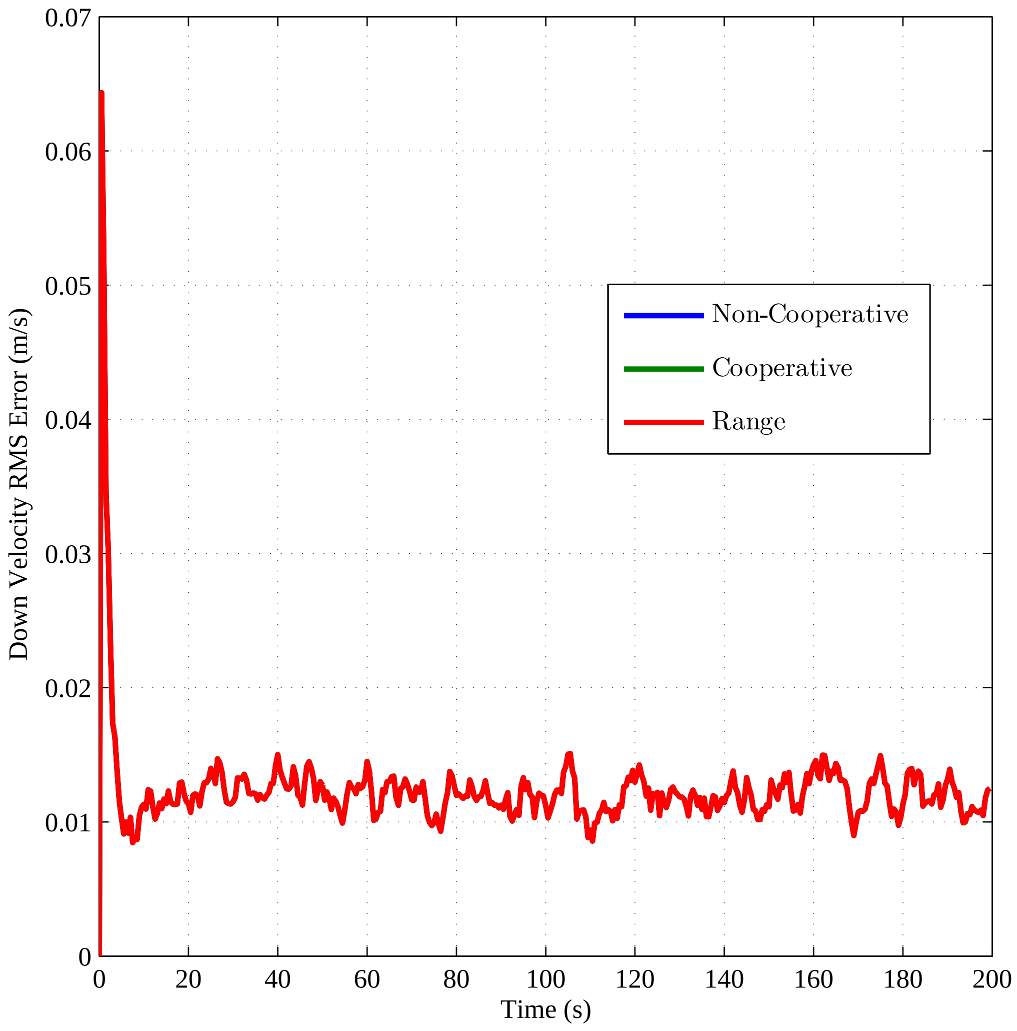


Figure A.7: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

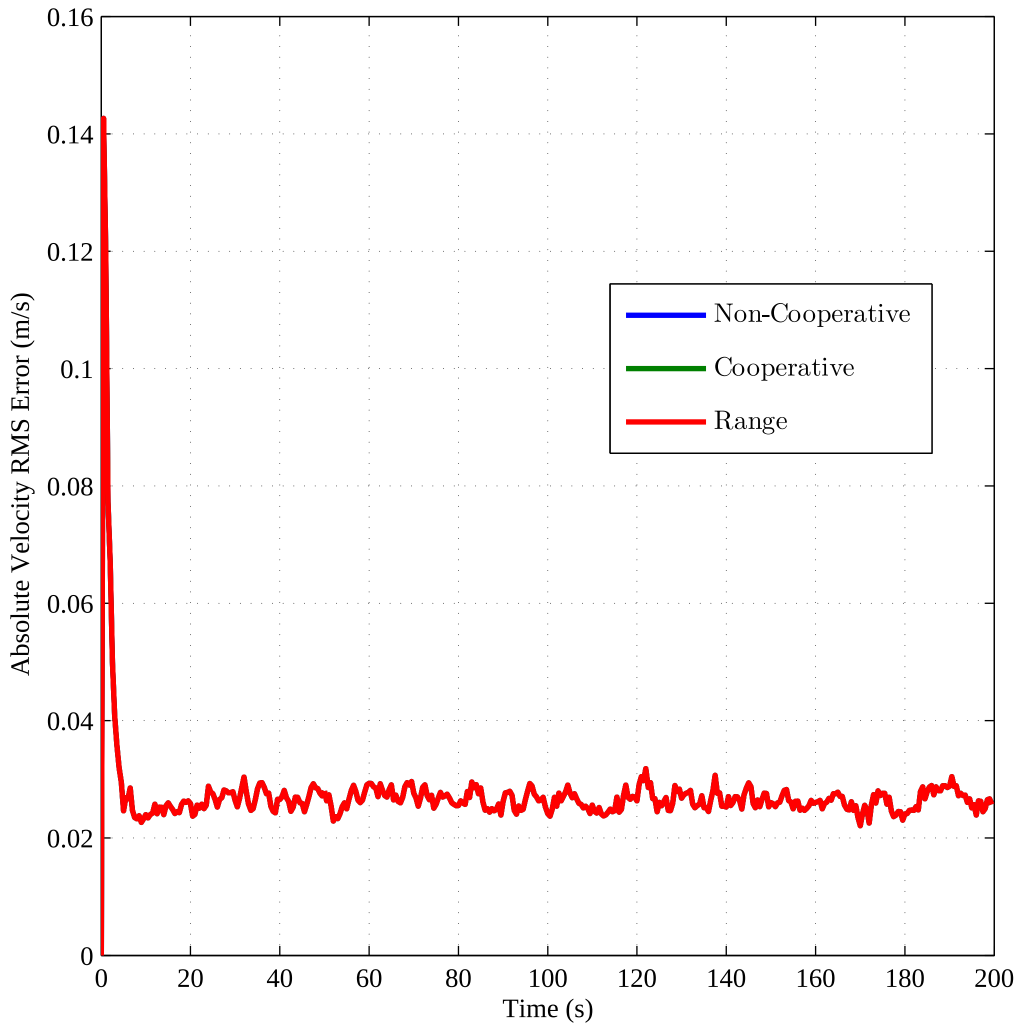


Figure A.8: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute velocity error results . The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

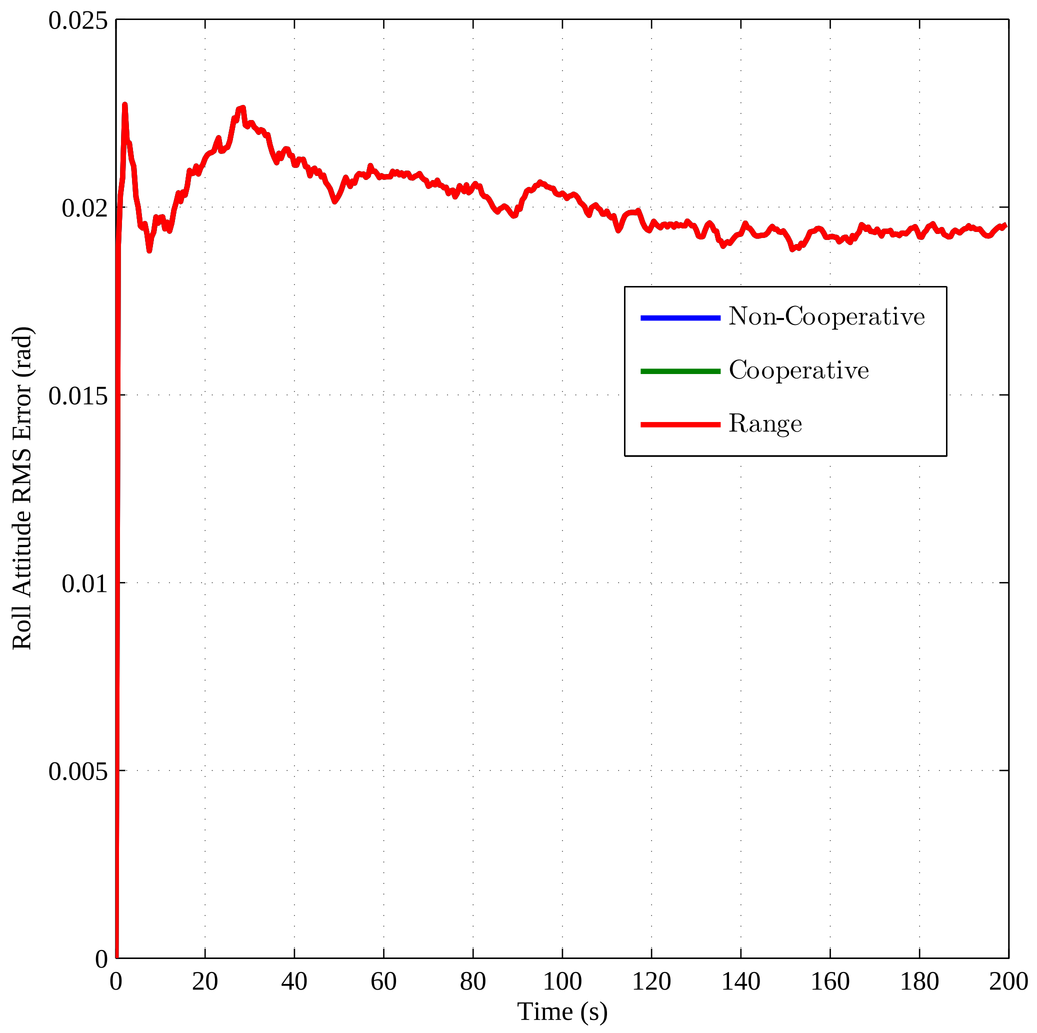


Figure A.9: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

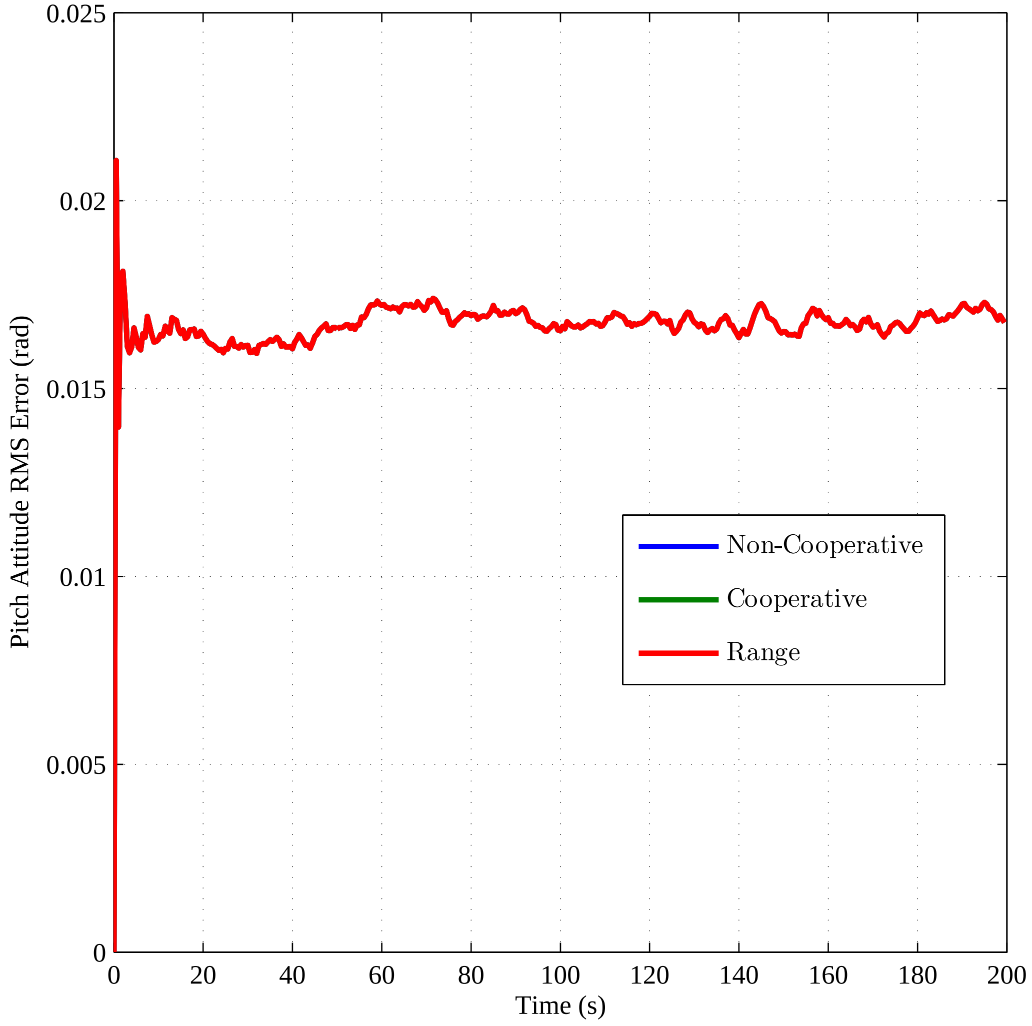


Figure A.10: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

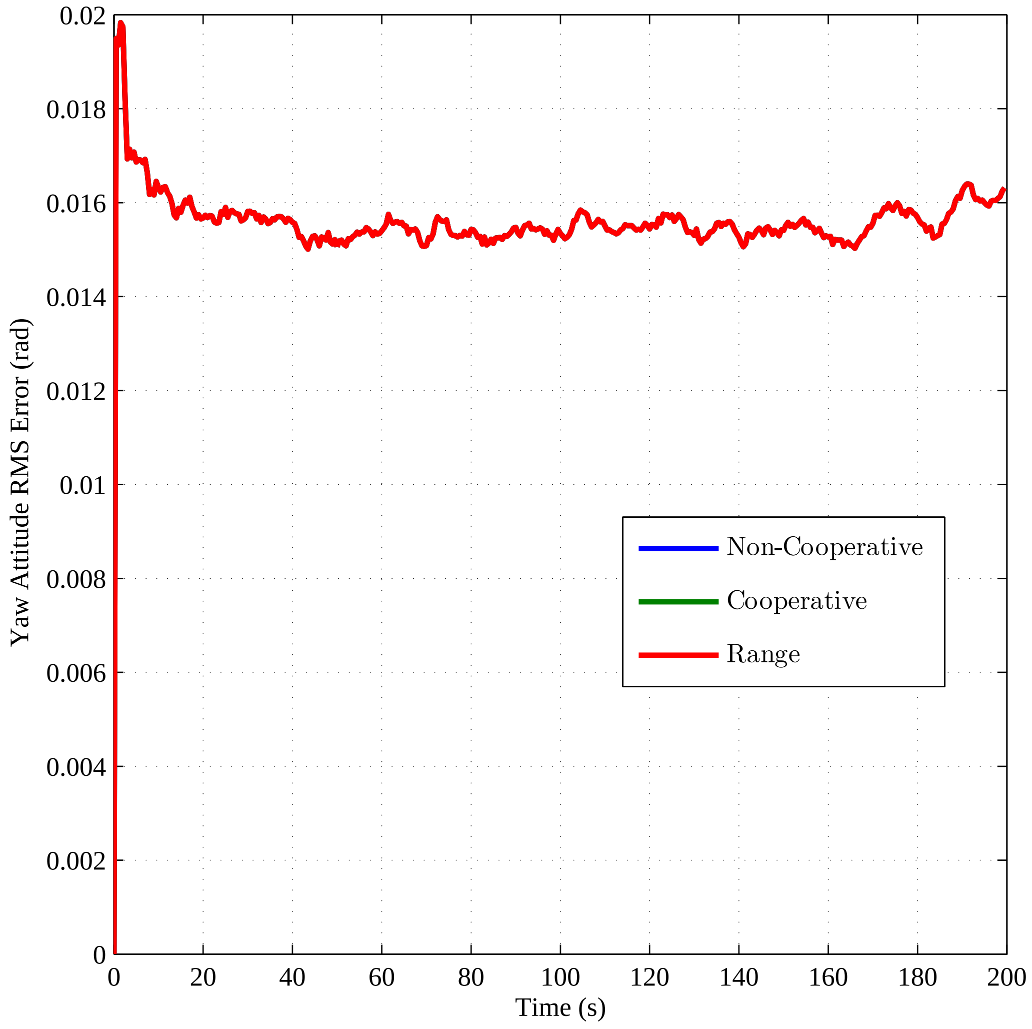


Figure A.11: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the post-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red). Note: the three system results presented here are identical due to the method used to implement the primary filter.

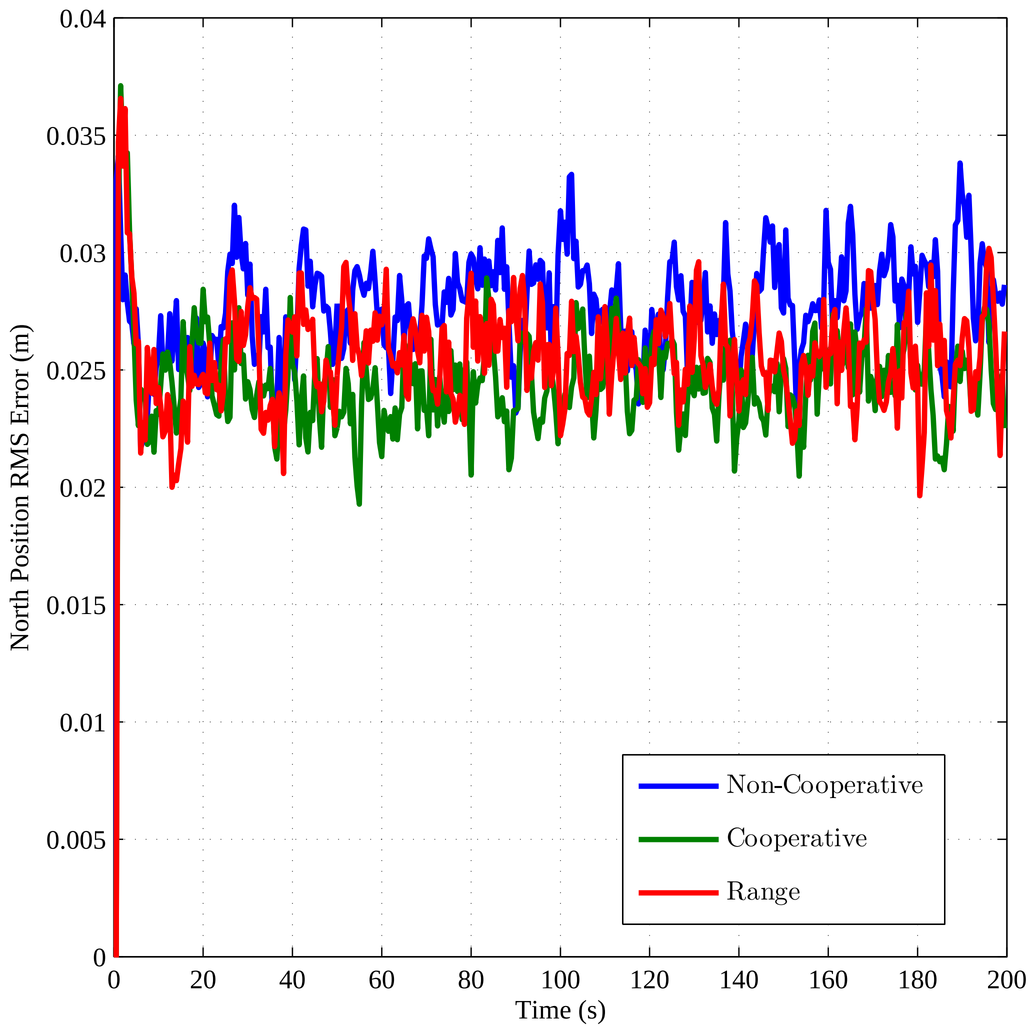


Figure B.1: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

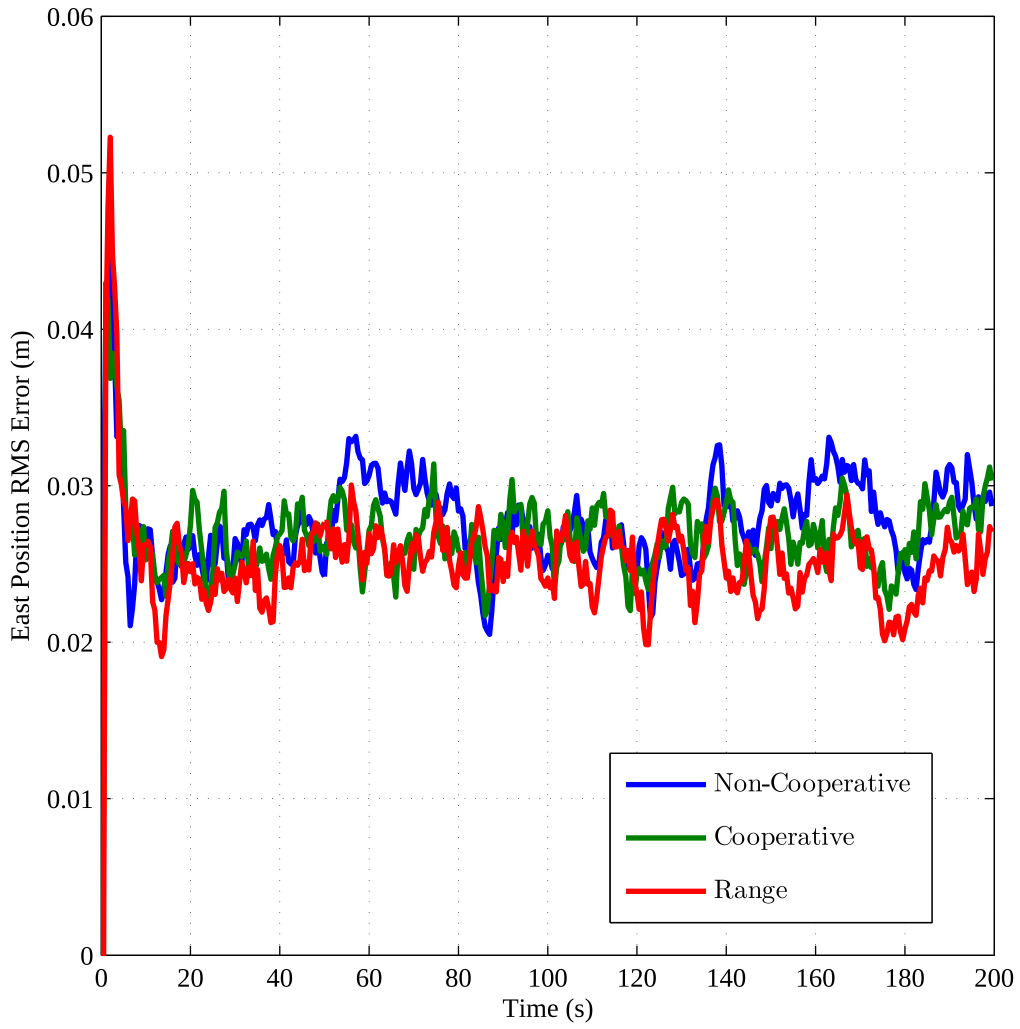


Figure B.2: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

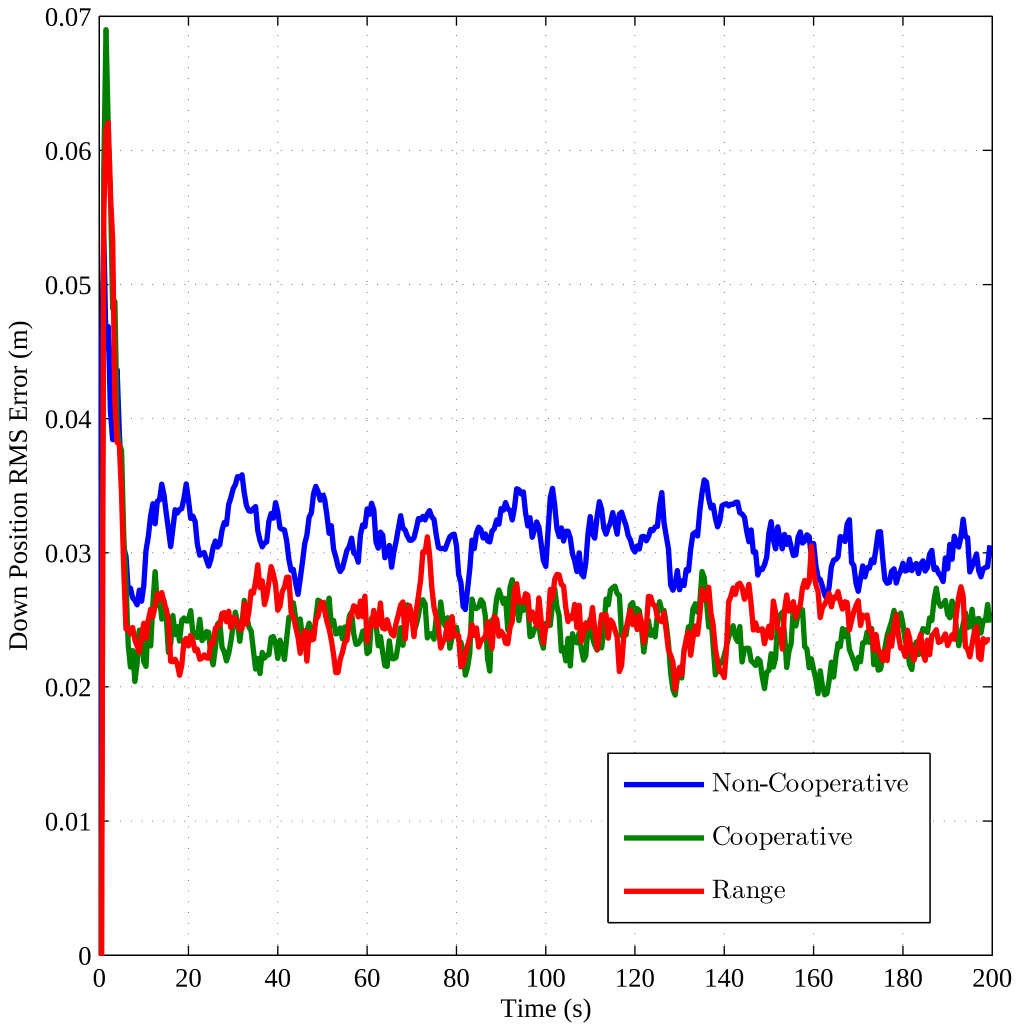


Figure B.3: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

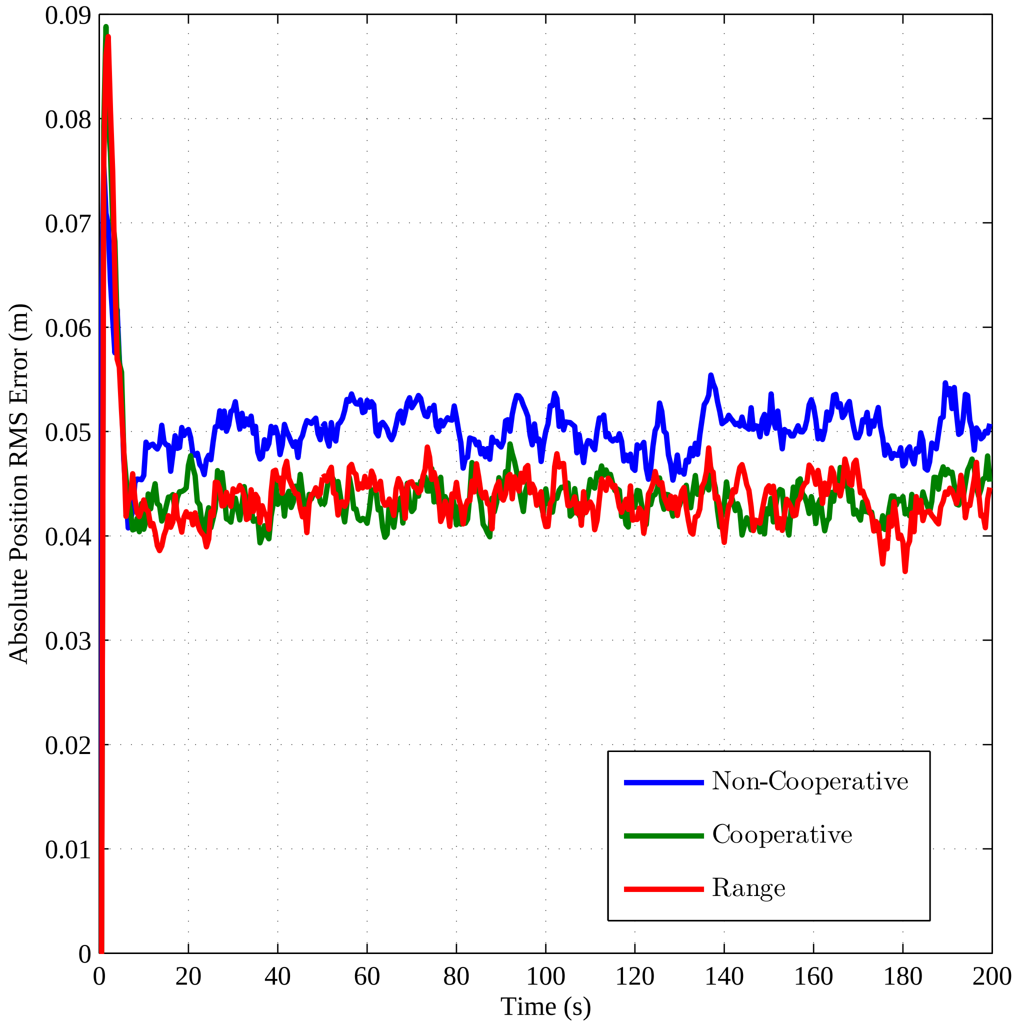


Figure B.4: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute position error results. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

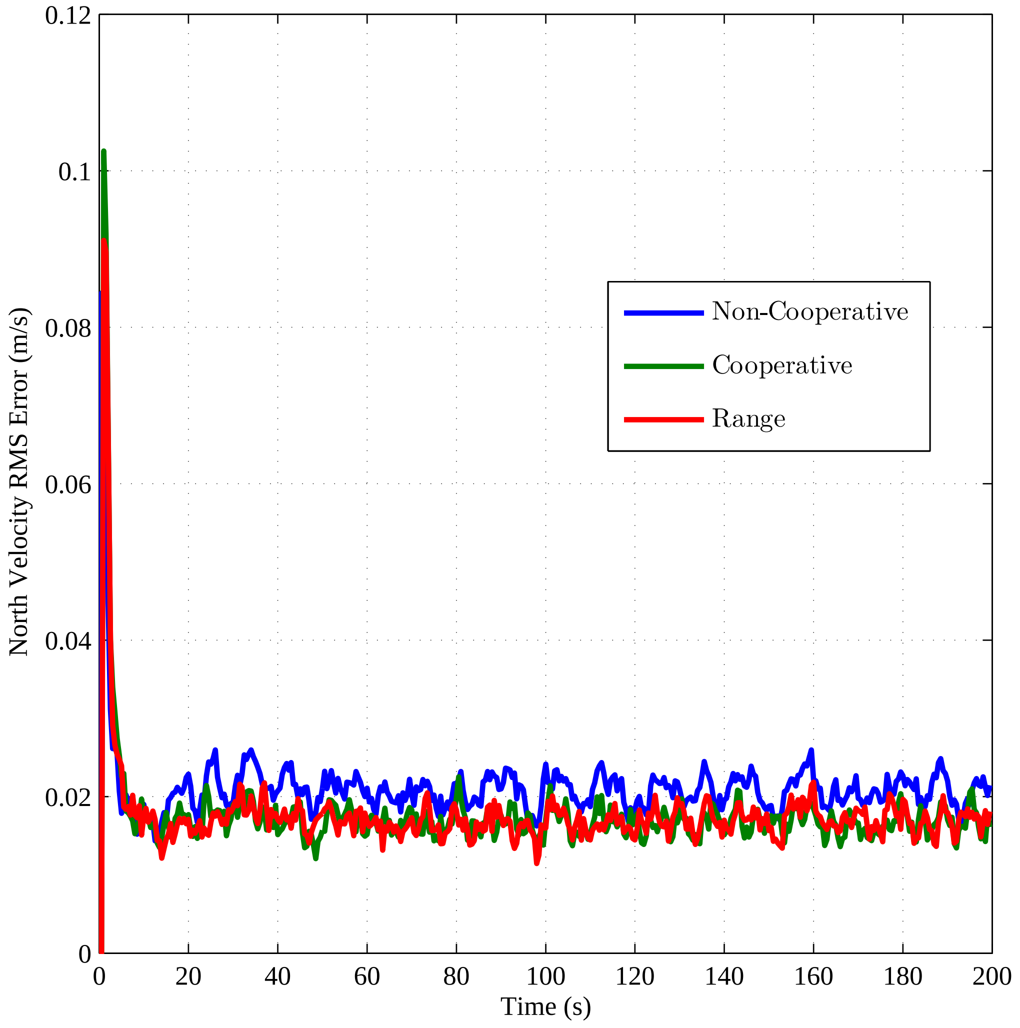


Figure B.5: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

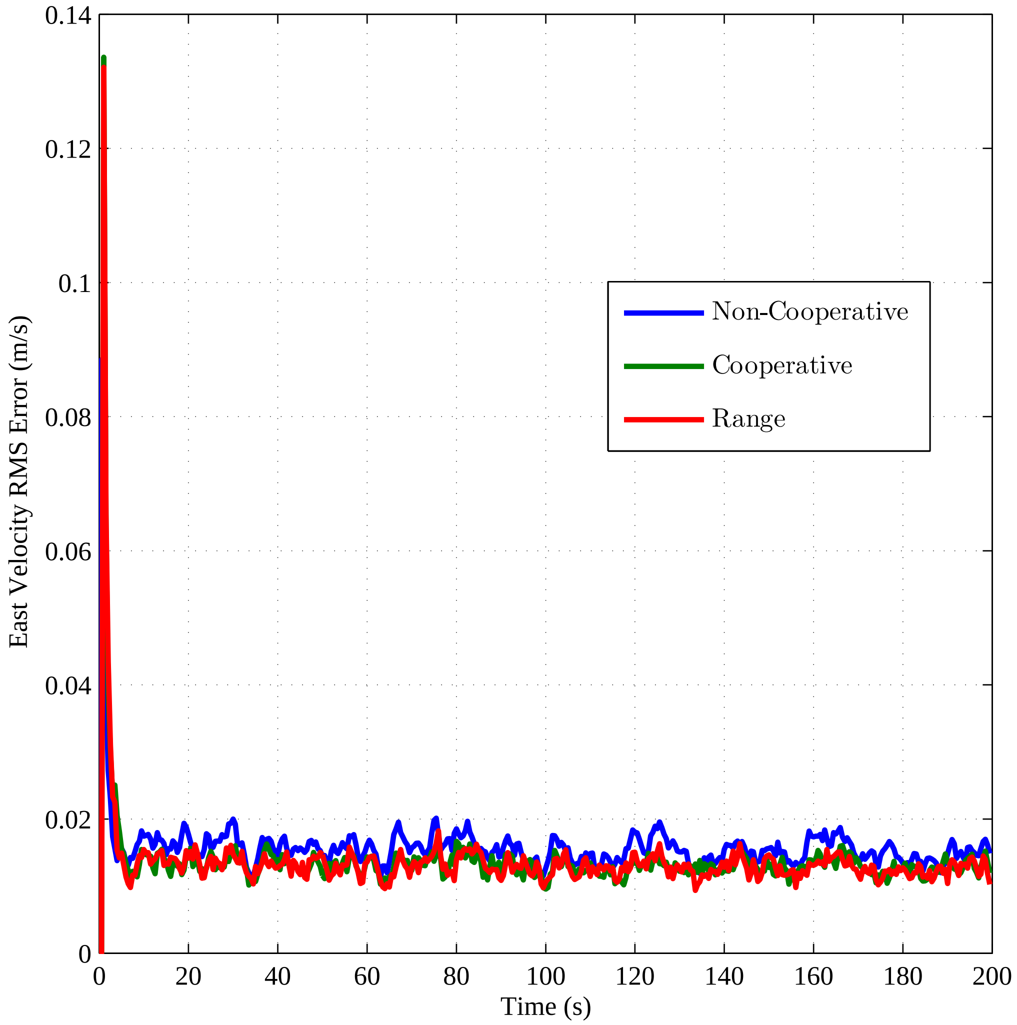


Figure B.6: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

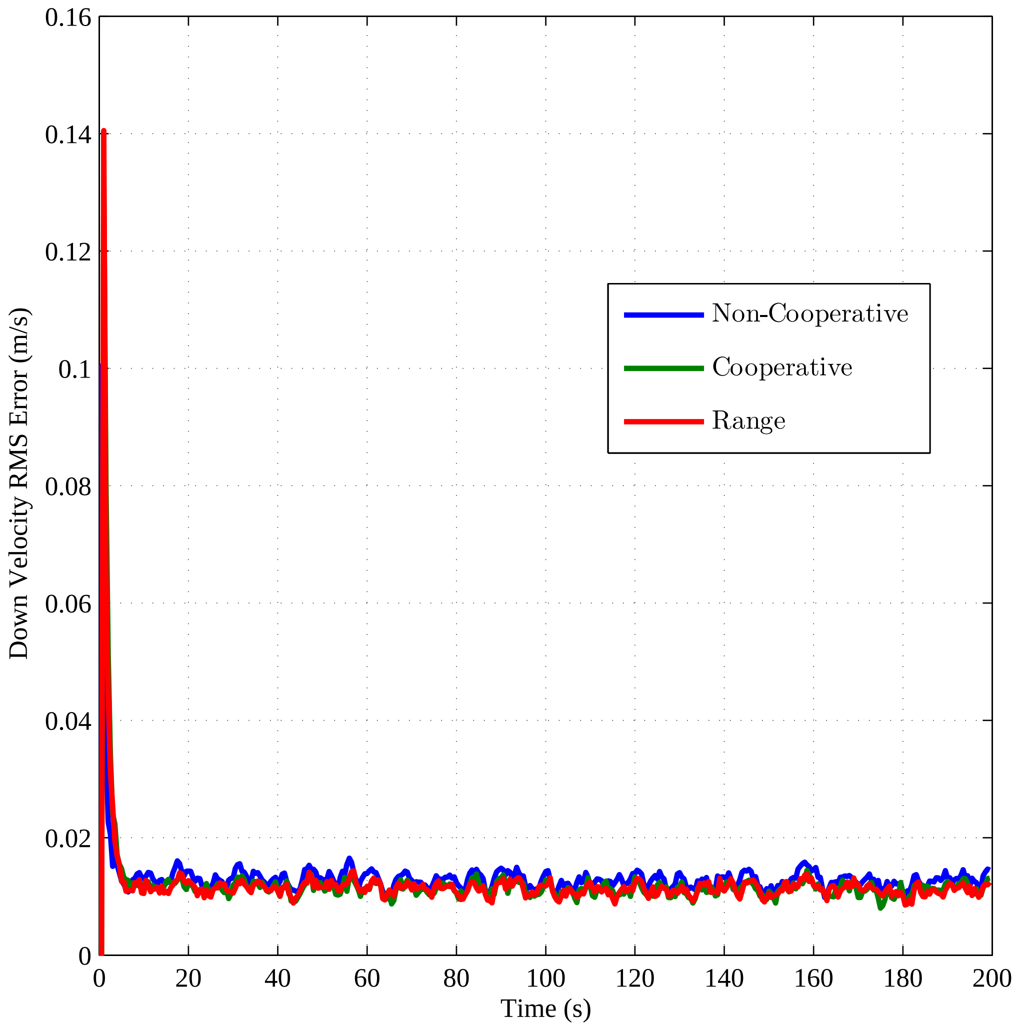


Figure B.7: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

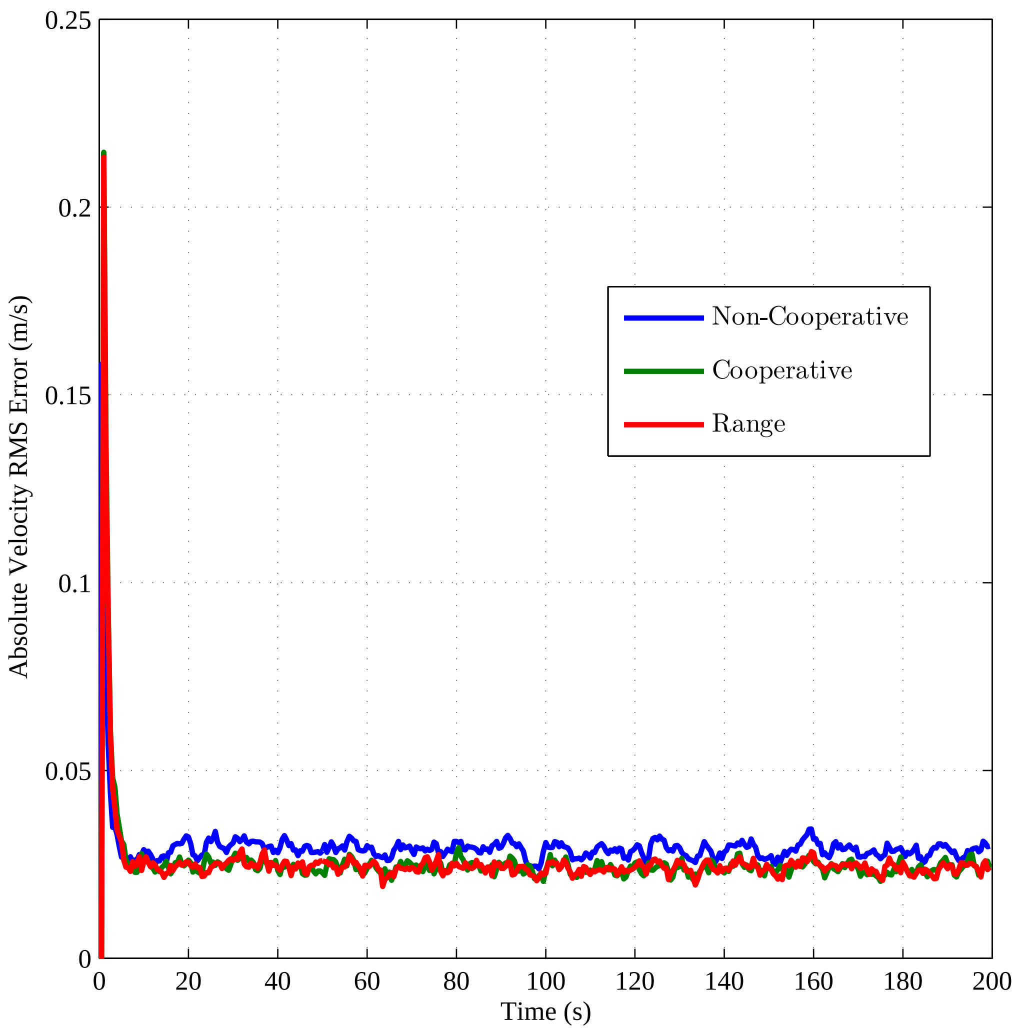


Figure B.8: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute velocity error results . The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

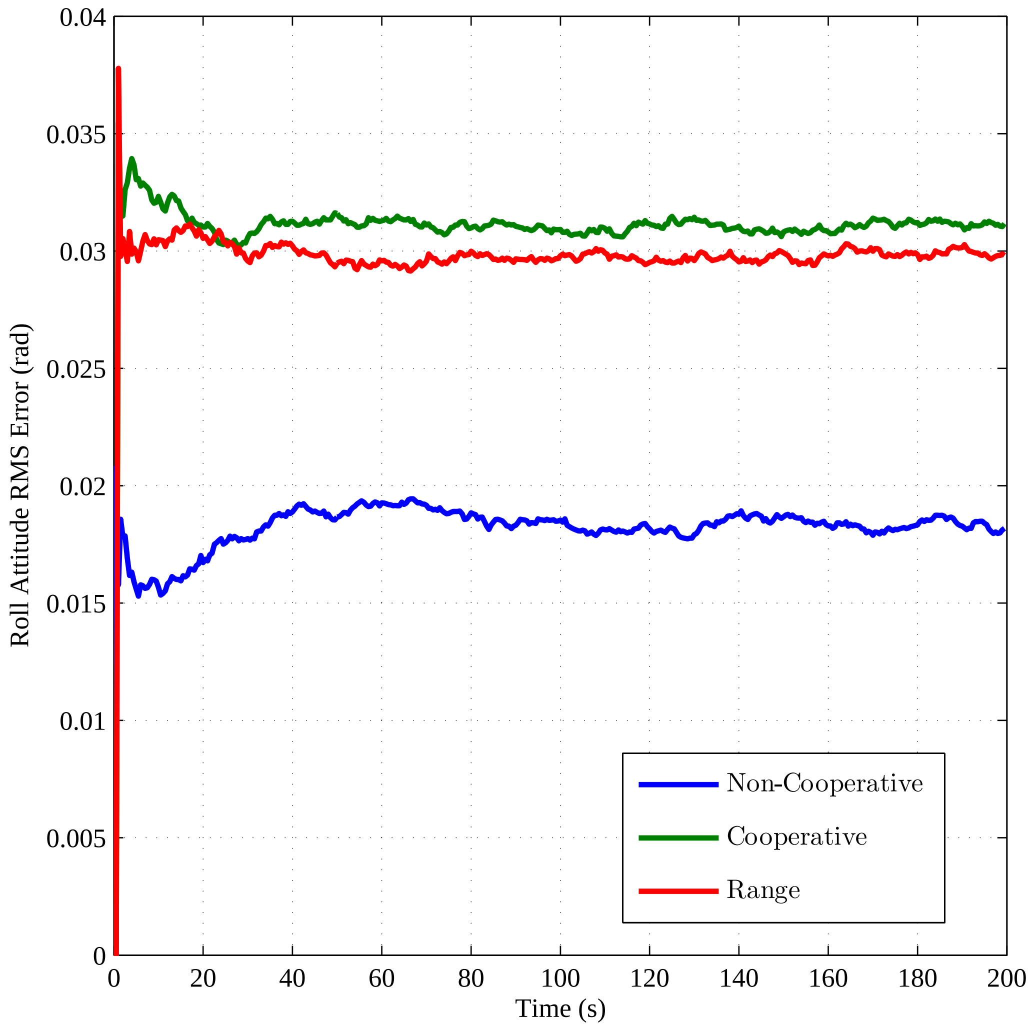


Figure B.9: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

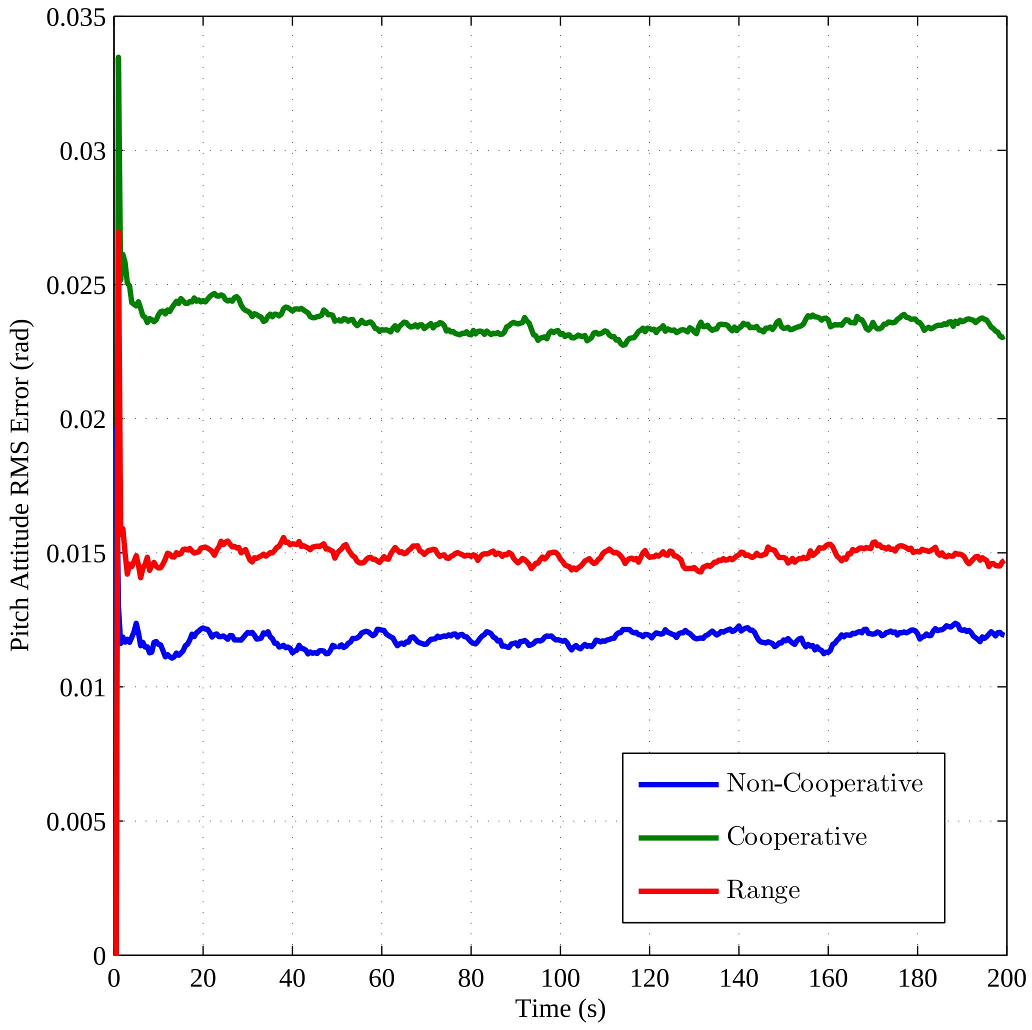


Figure B.10: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

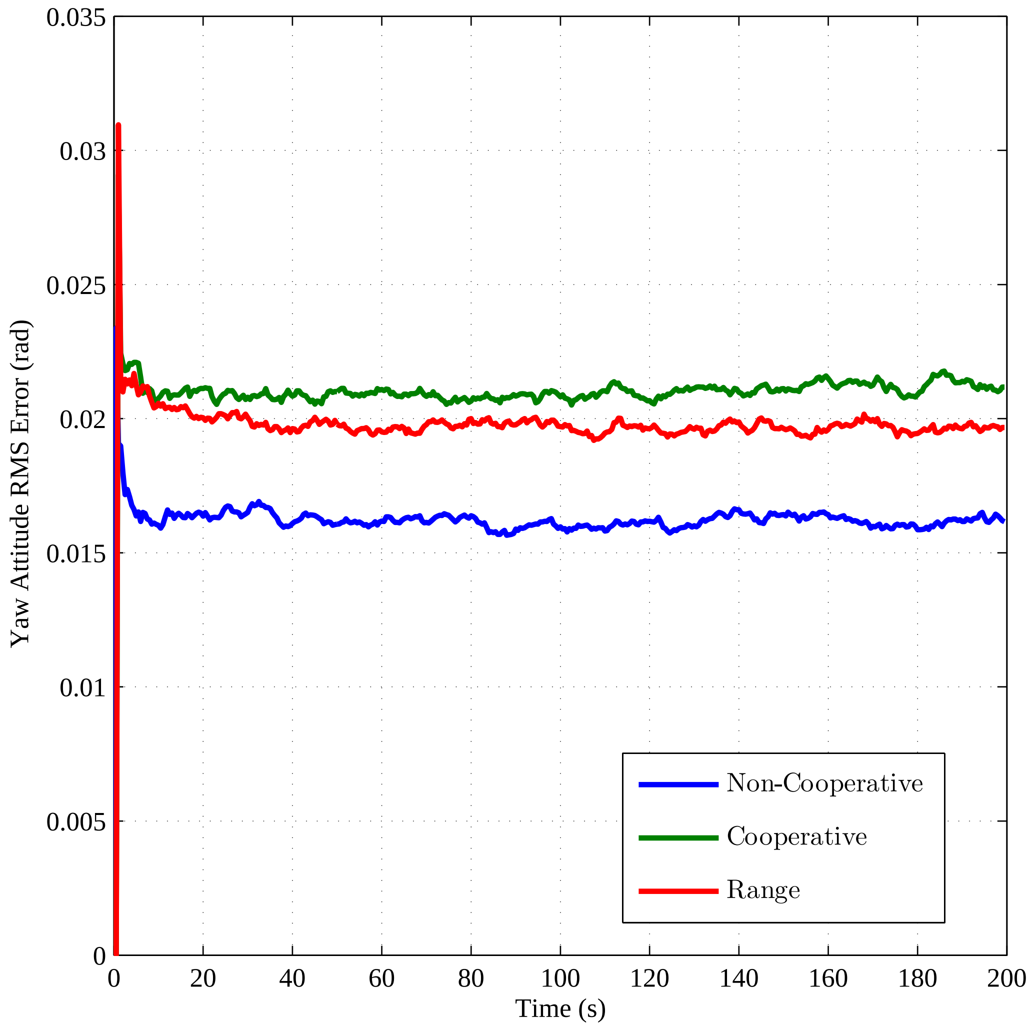


Figure B.11: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the first of two stationary platforms simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

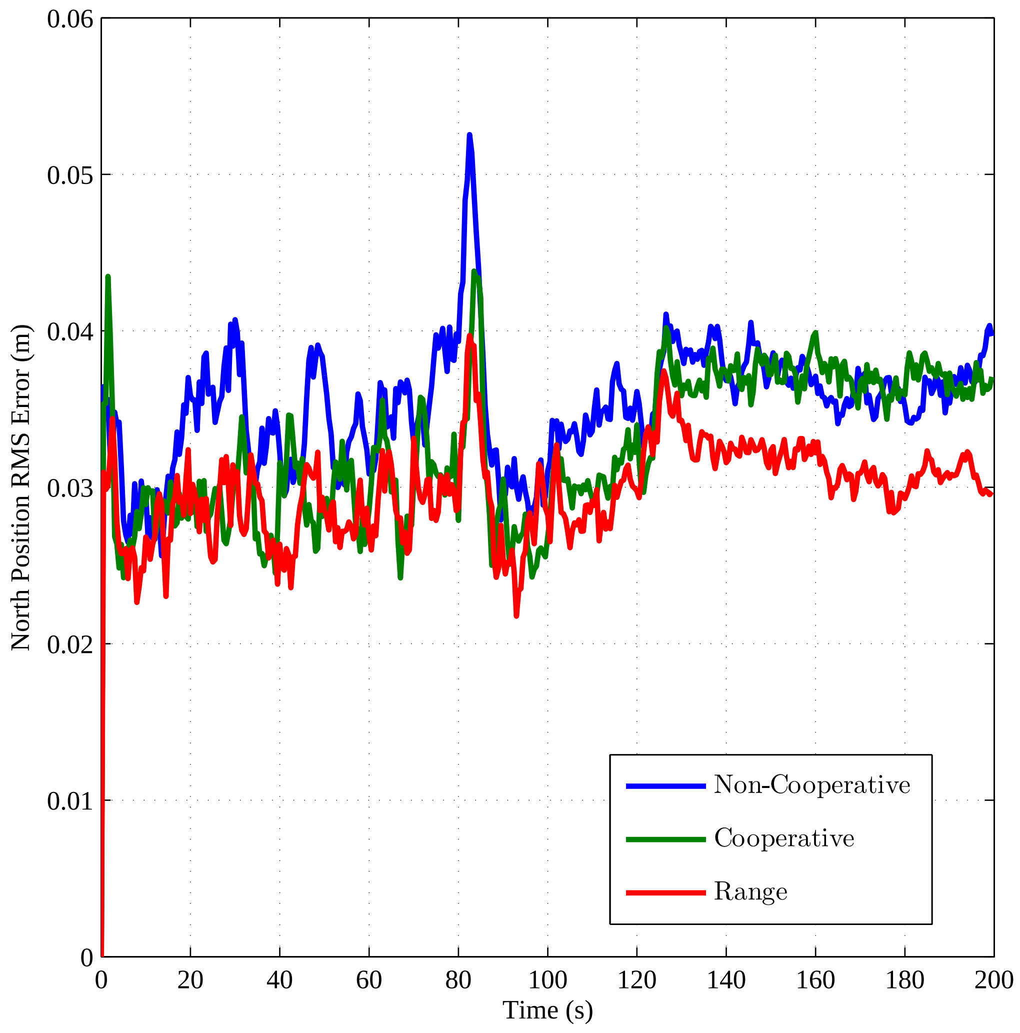


Figure C.1: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

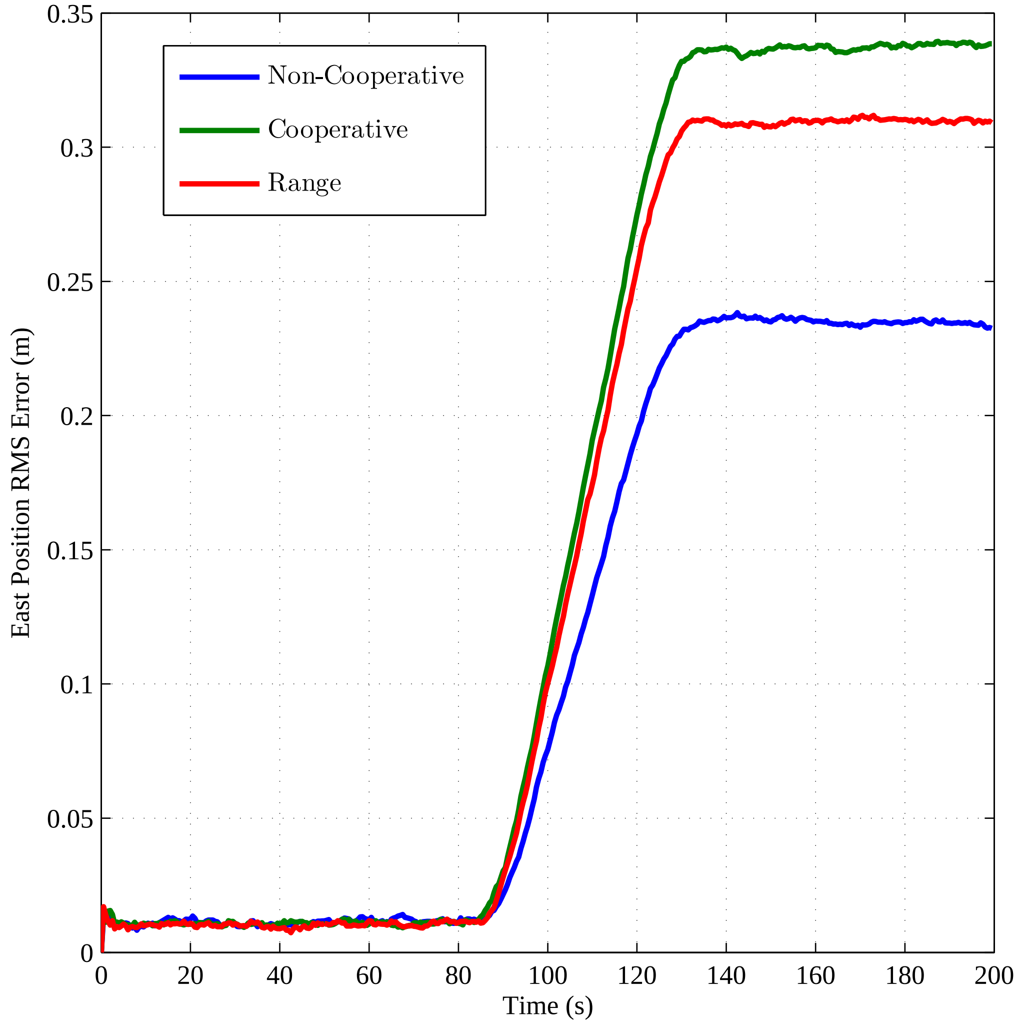


Figure C.2: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the east axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

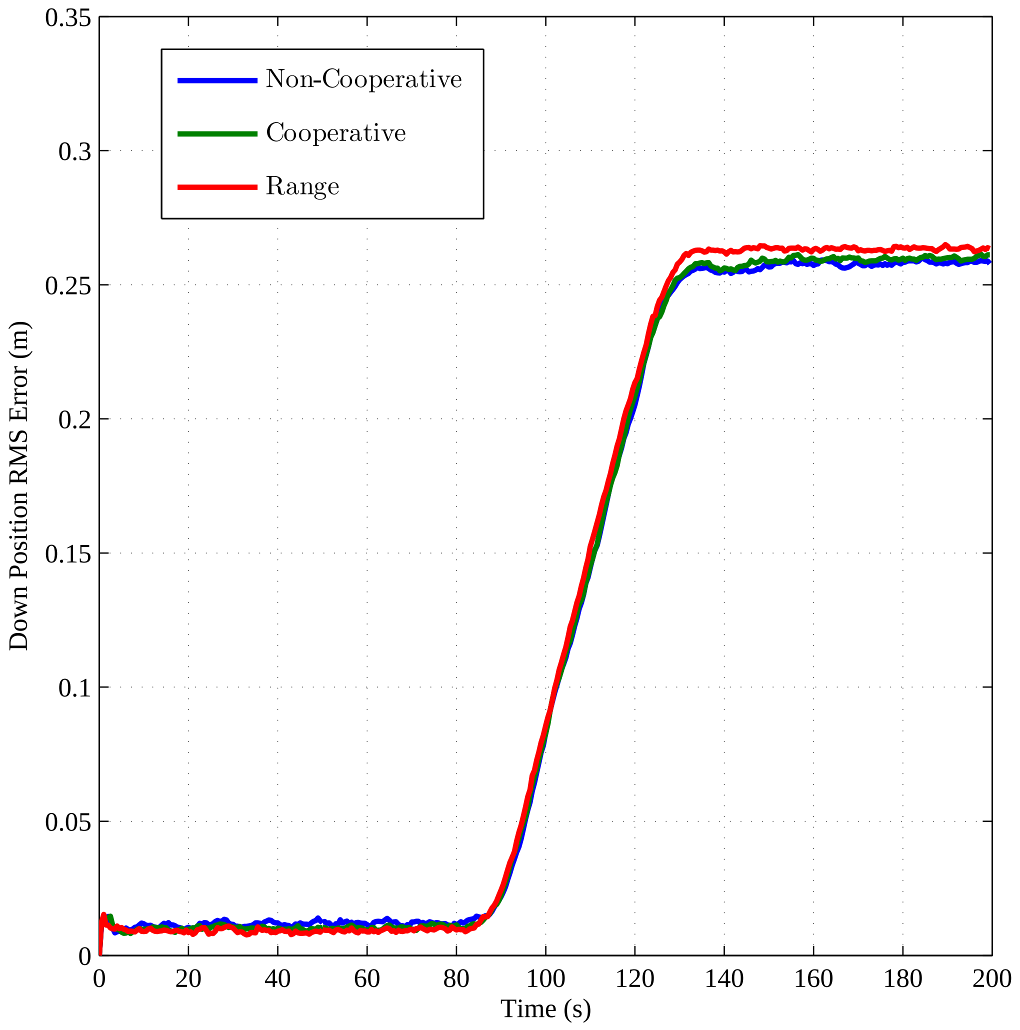


Figure C.3: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a $25\,m$ hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

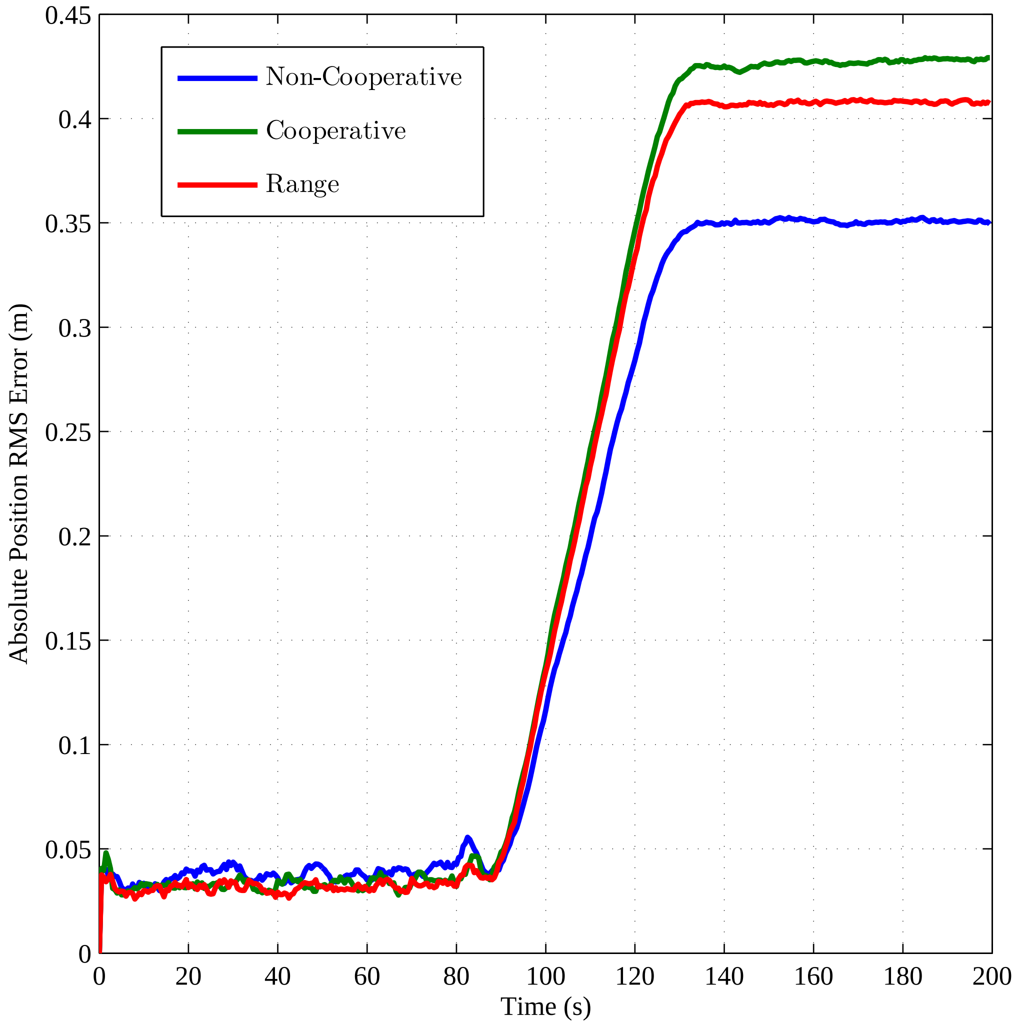


Figure C.4: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute position error results. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the prescaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

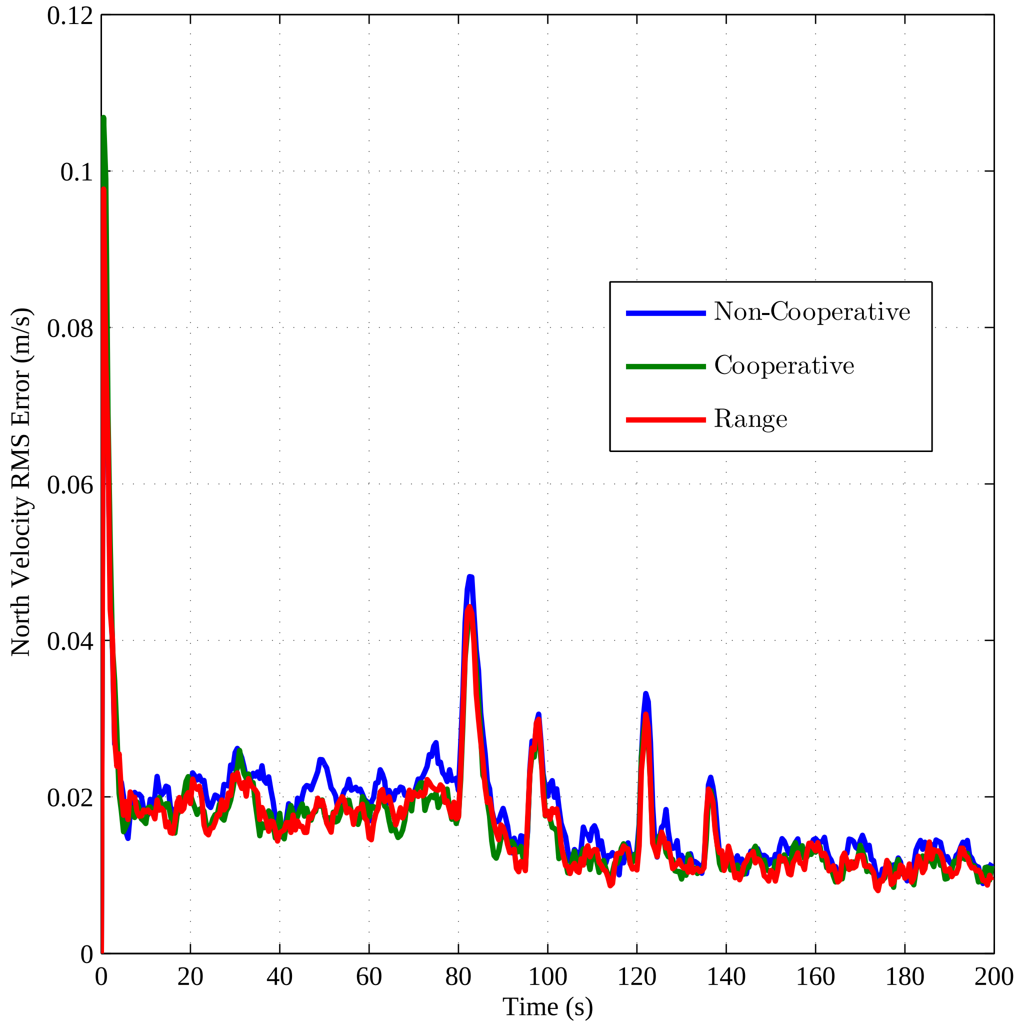


Figure C.5: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

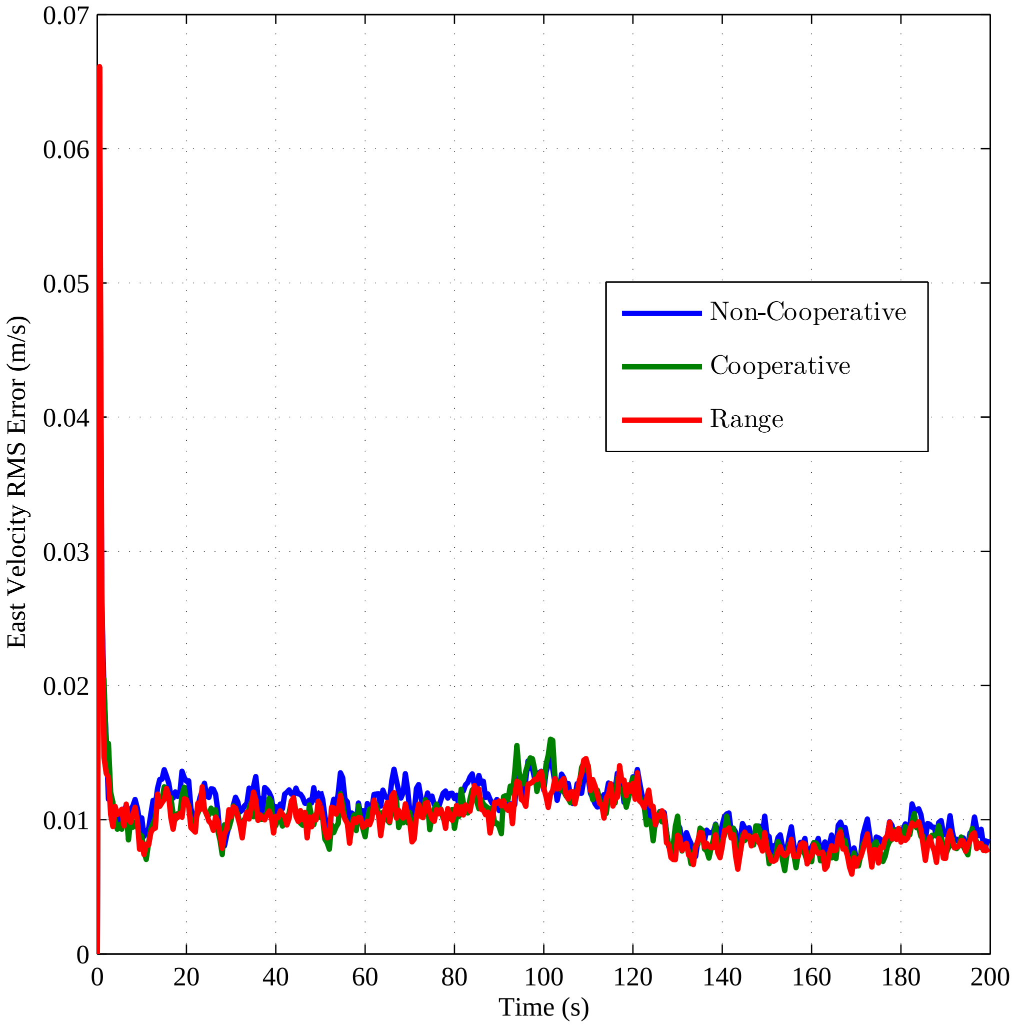


Figure C.6: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the east axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a $25\,m$ hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

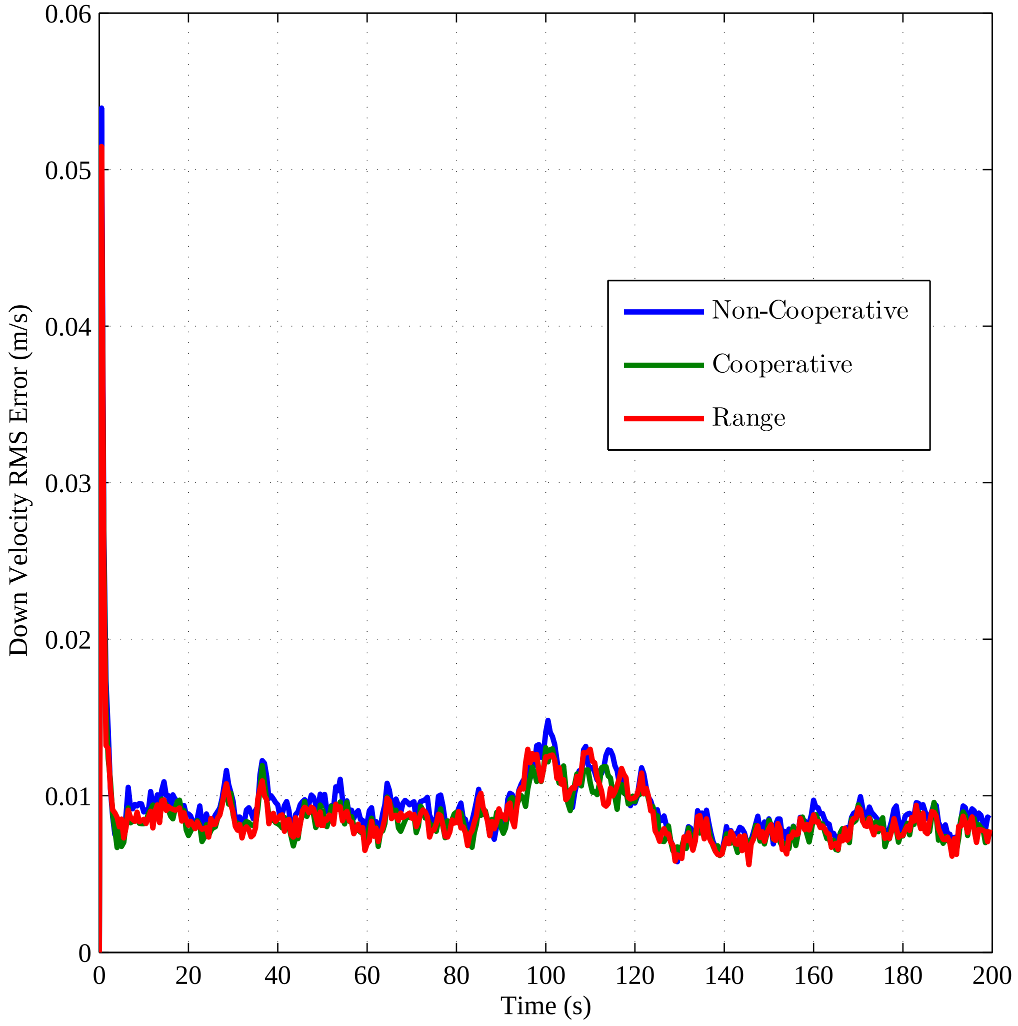


Figure C.7: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

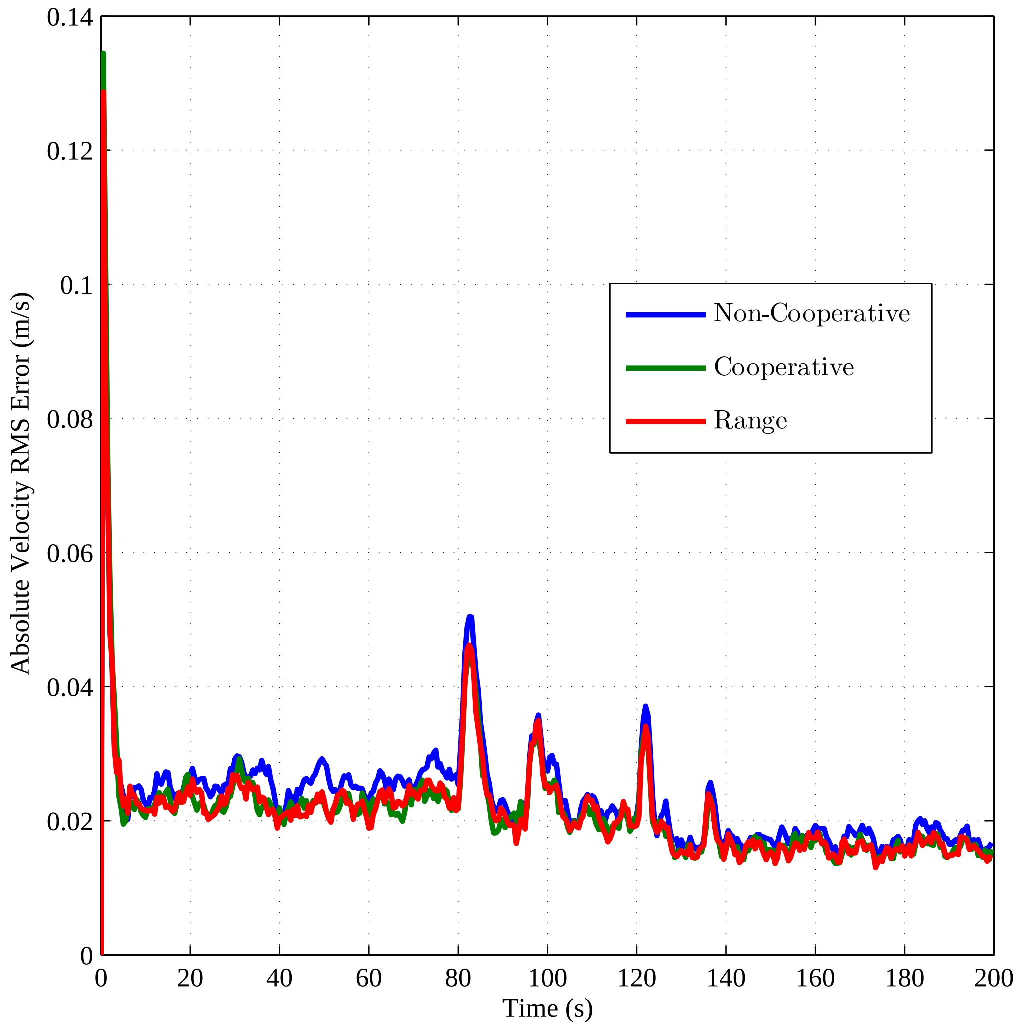


Figure C.8: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute velocity error results . The results shown are for the first of two platforms, traveling along a $25\,m$ hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

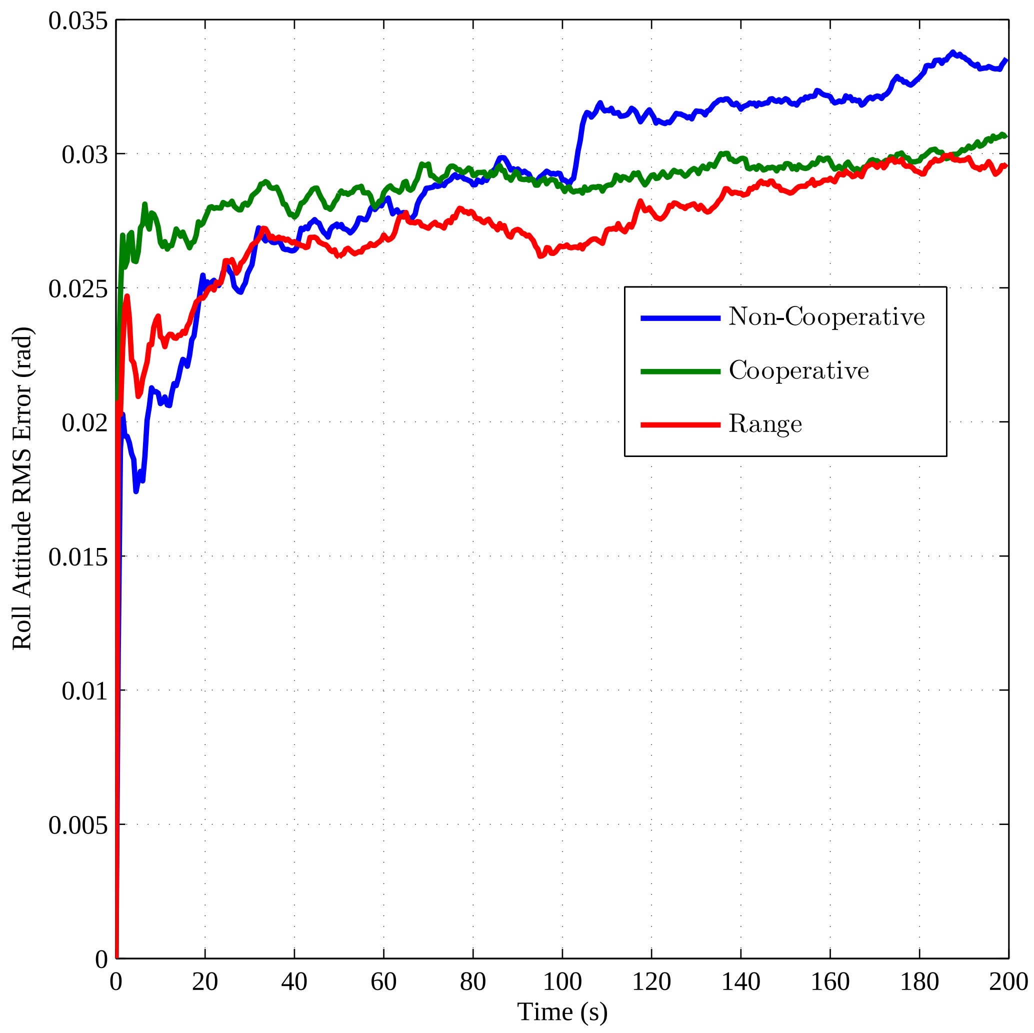


Figure C.9: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

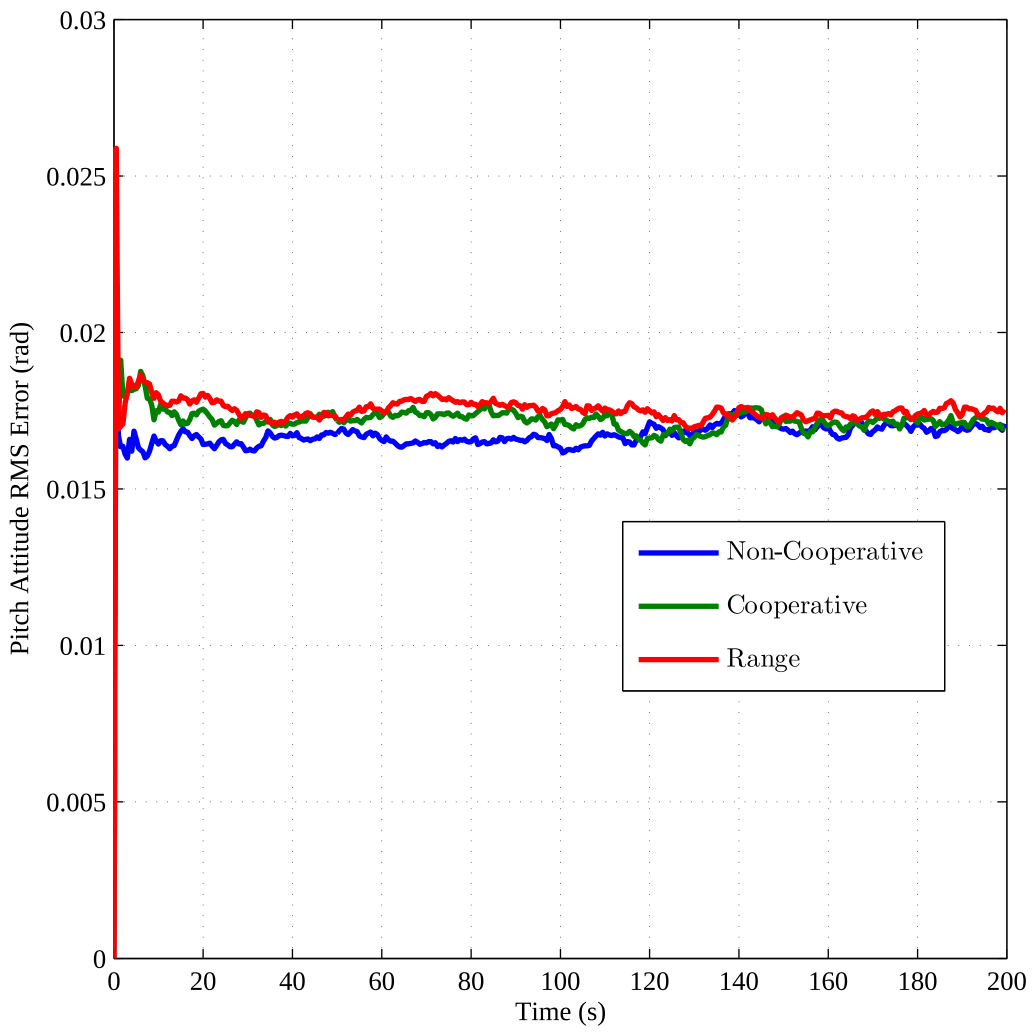


Figure C.10: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

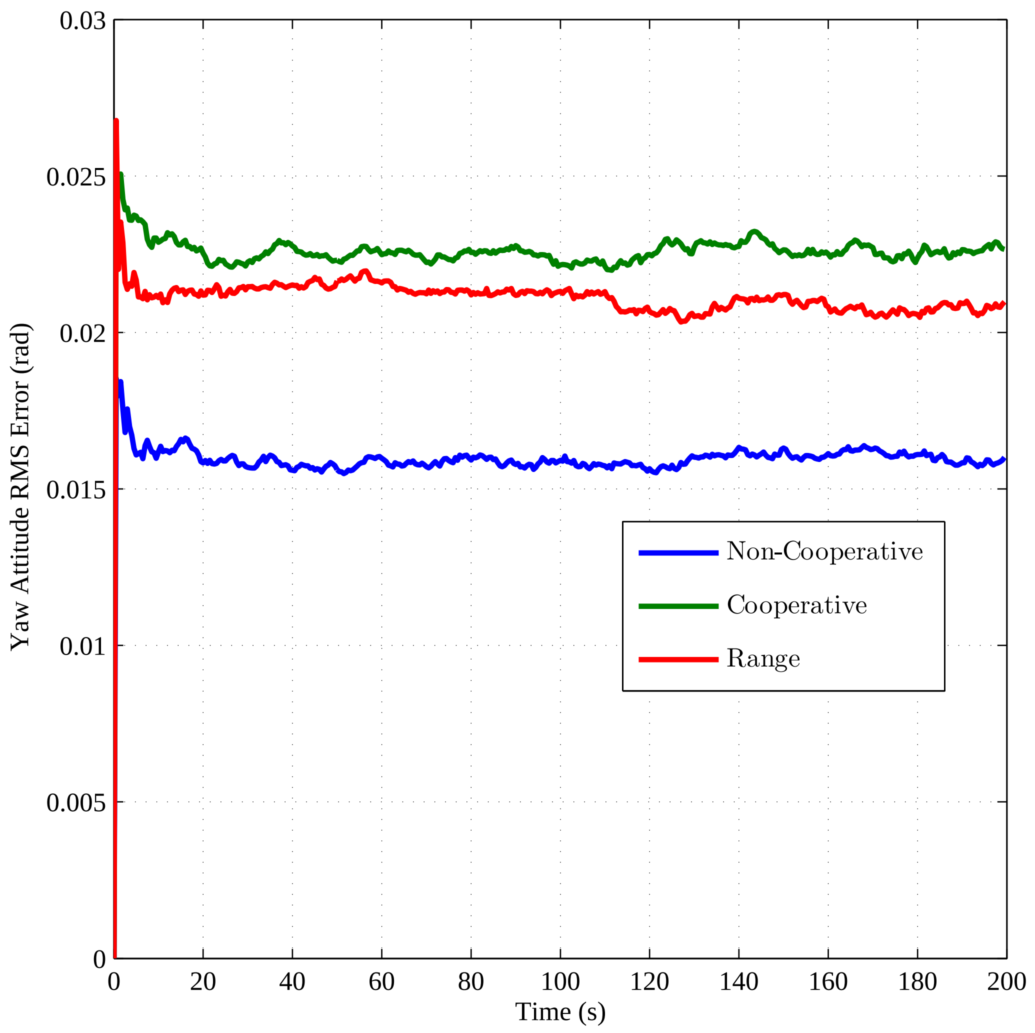


Figure C.11: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along a 25 m hallway, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

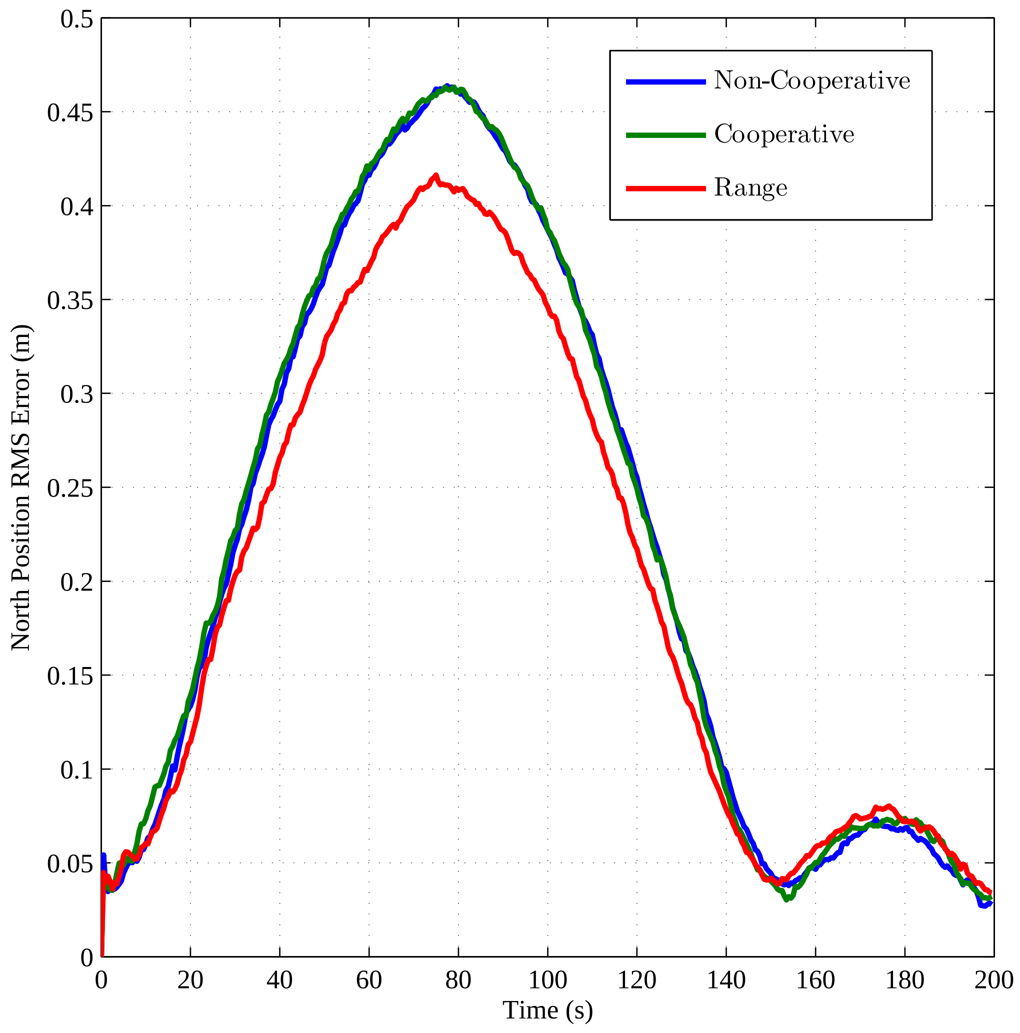


Figure D.1: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

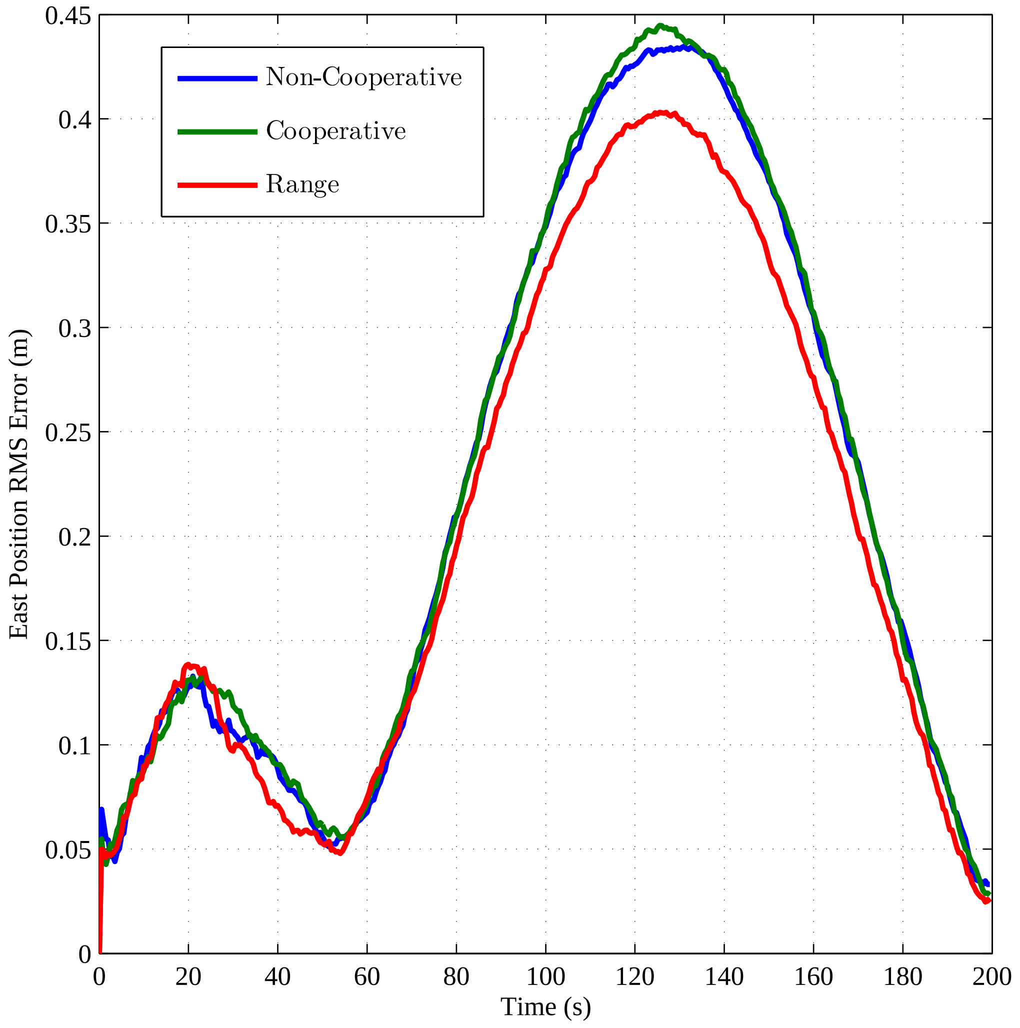


Figure D.2: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the east axis of the local navigation frame. The results shown are for the first of two platforms traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

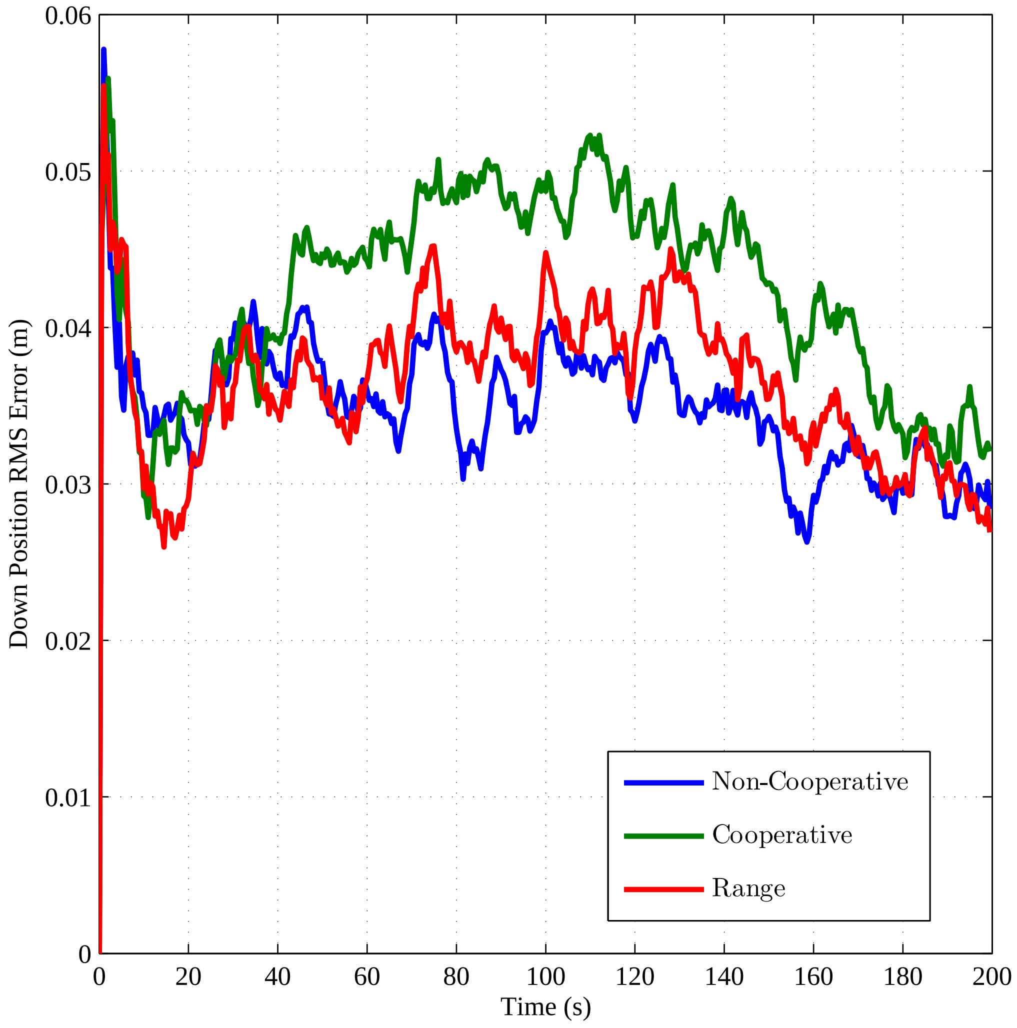


Figure D.3: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

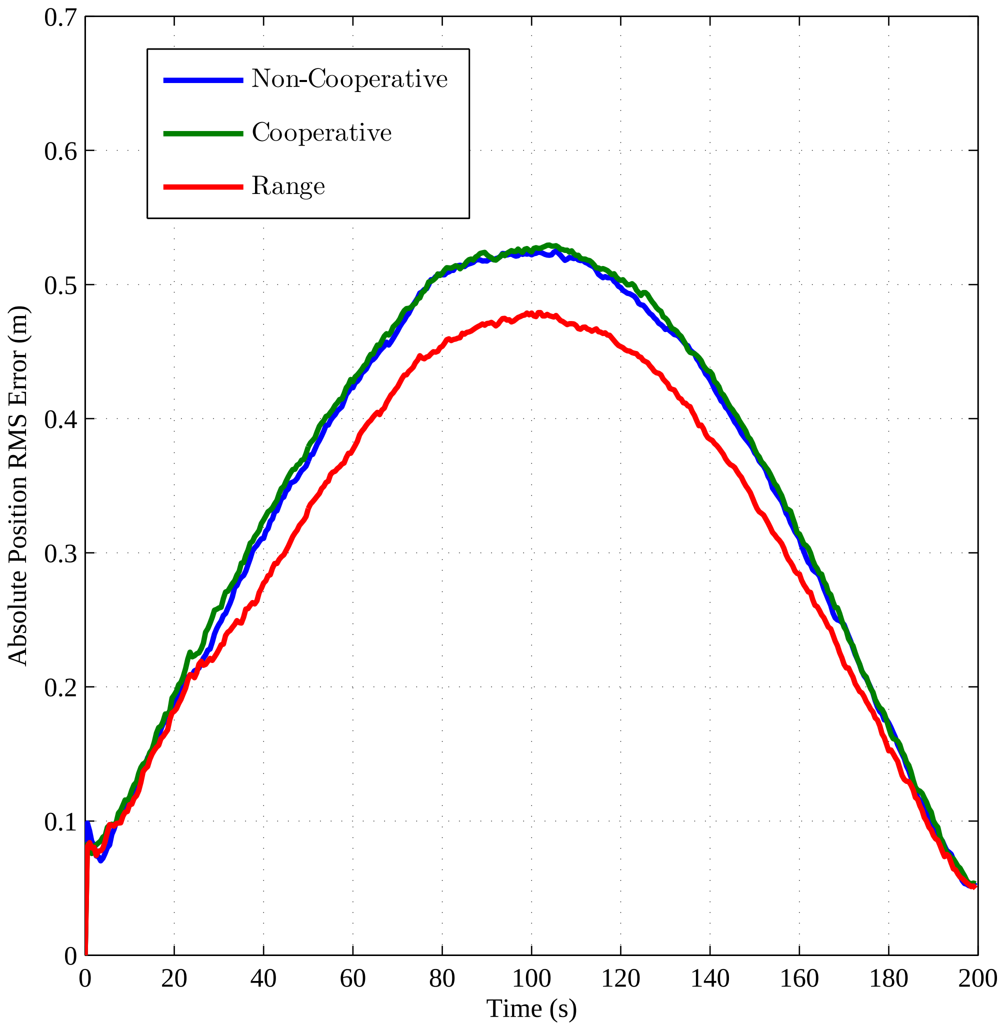


Figure D.4: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute position error results. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

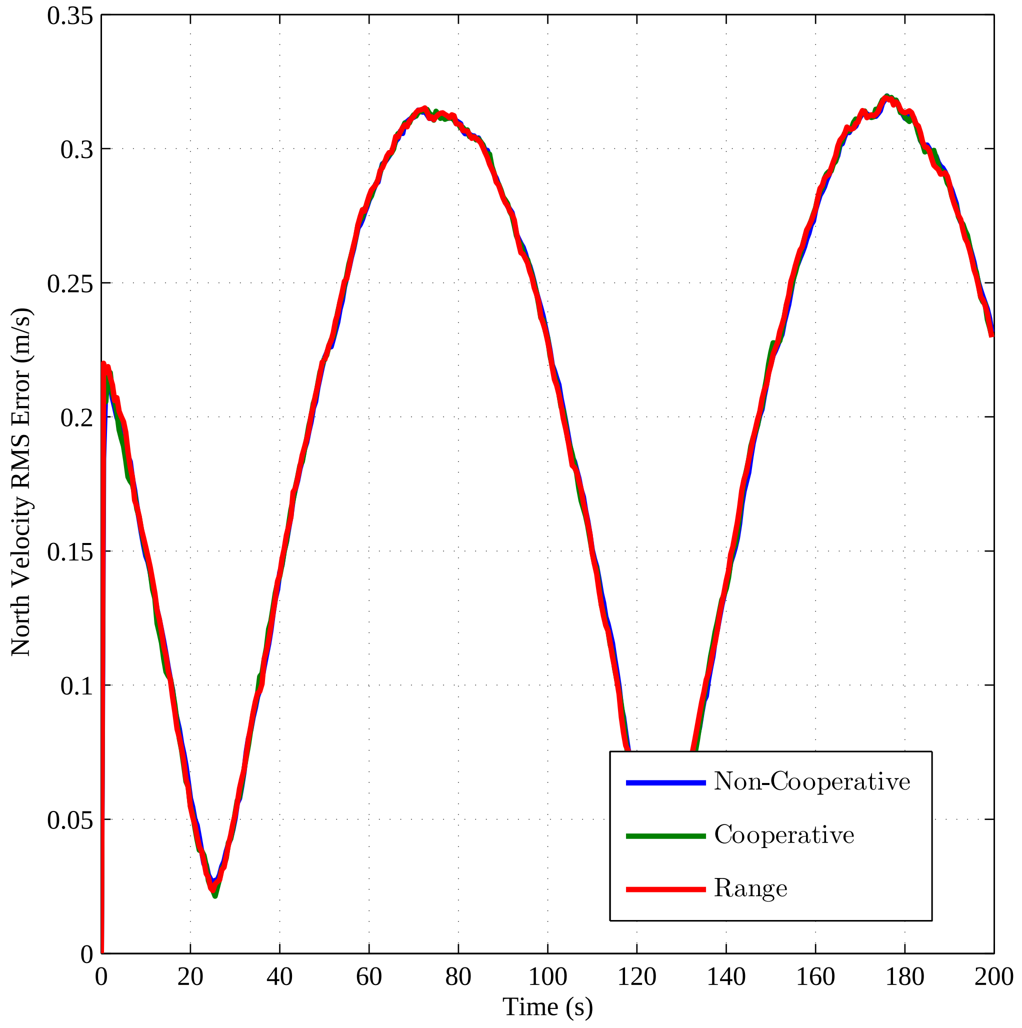


Figure D.5: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

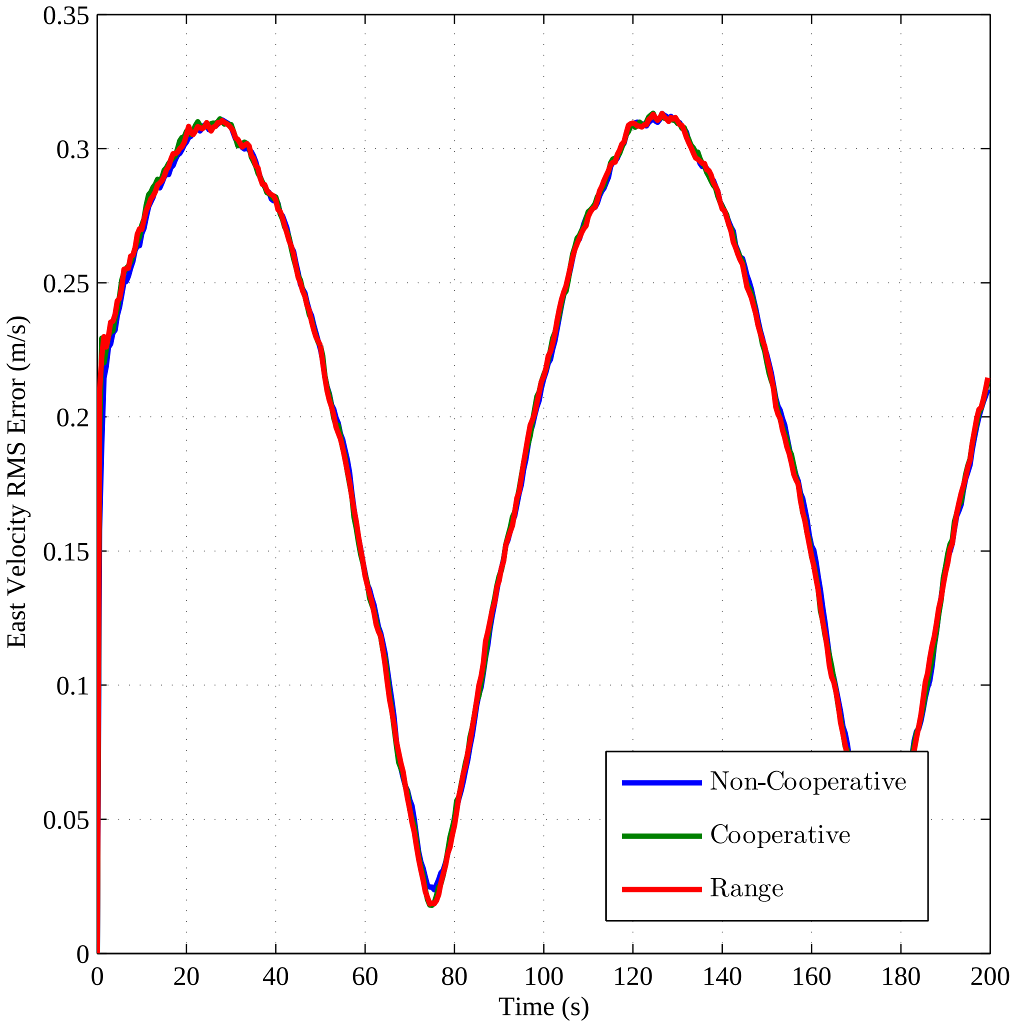


Figure D.6: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the east axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

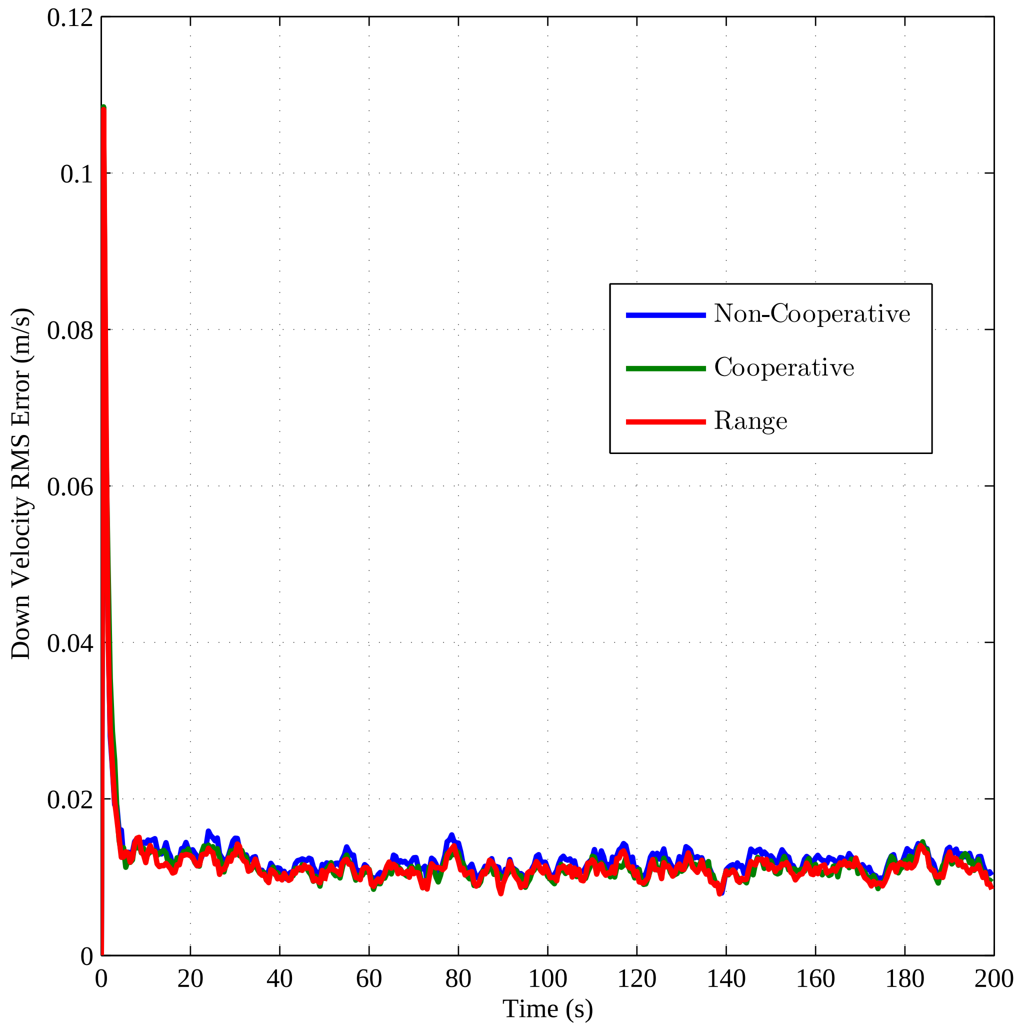


Figure D.7: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

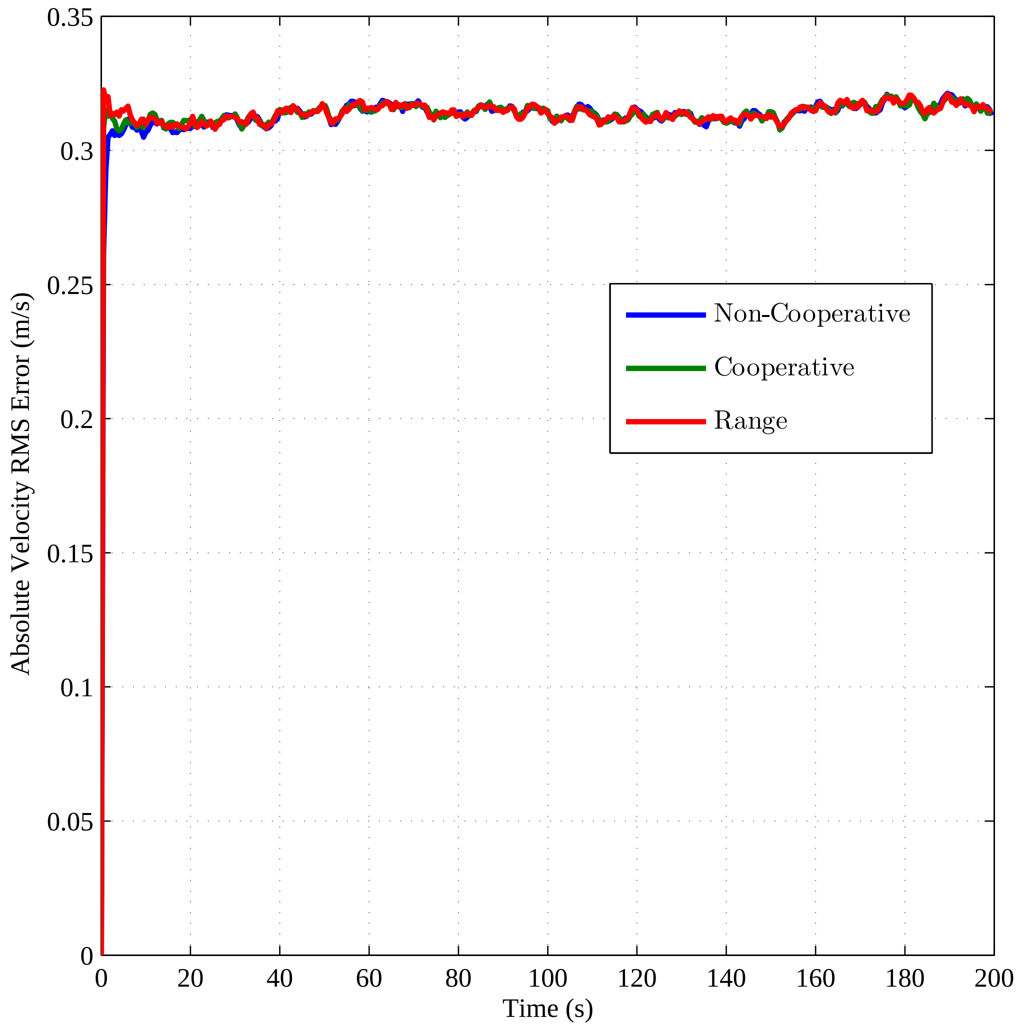


Figure D.8: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute velocity error results . The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

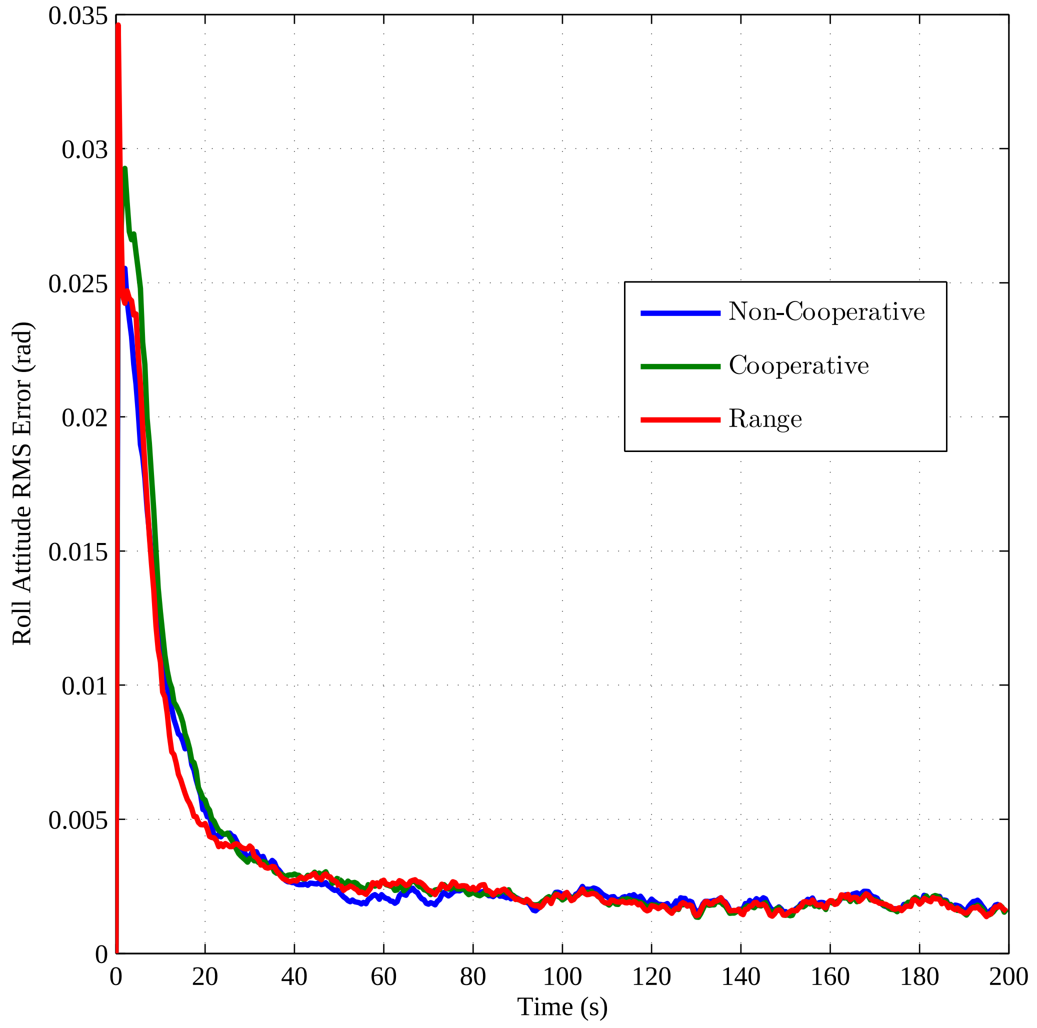


Figure D.9: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

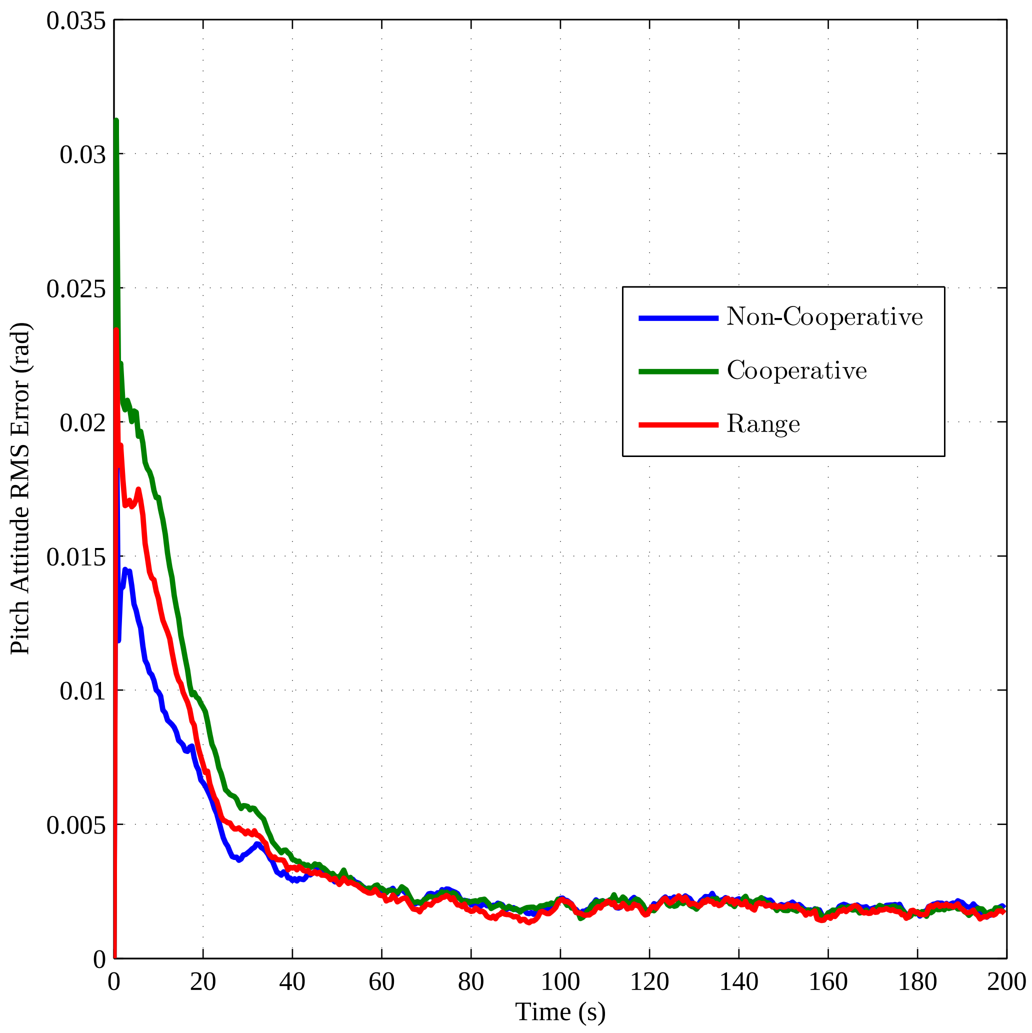


Figure D.10: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

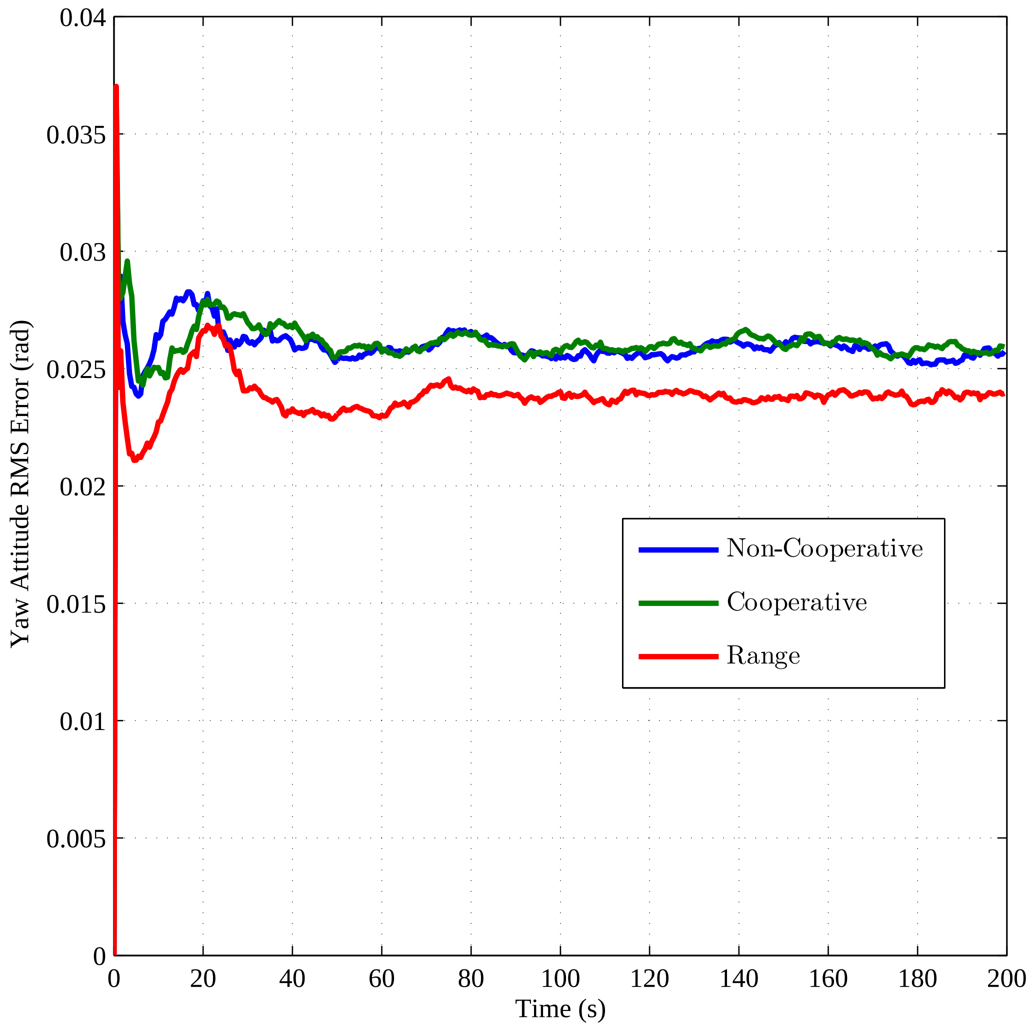


Figure D.11: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the first of two platforms, traveling along sideslip orbits around an object, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

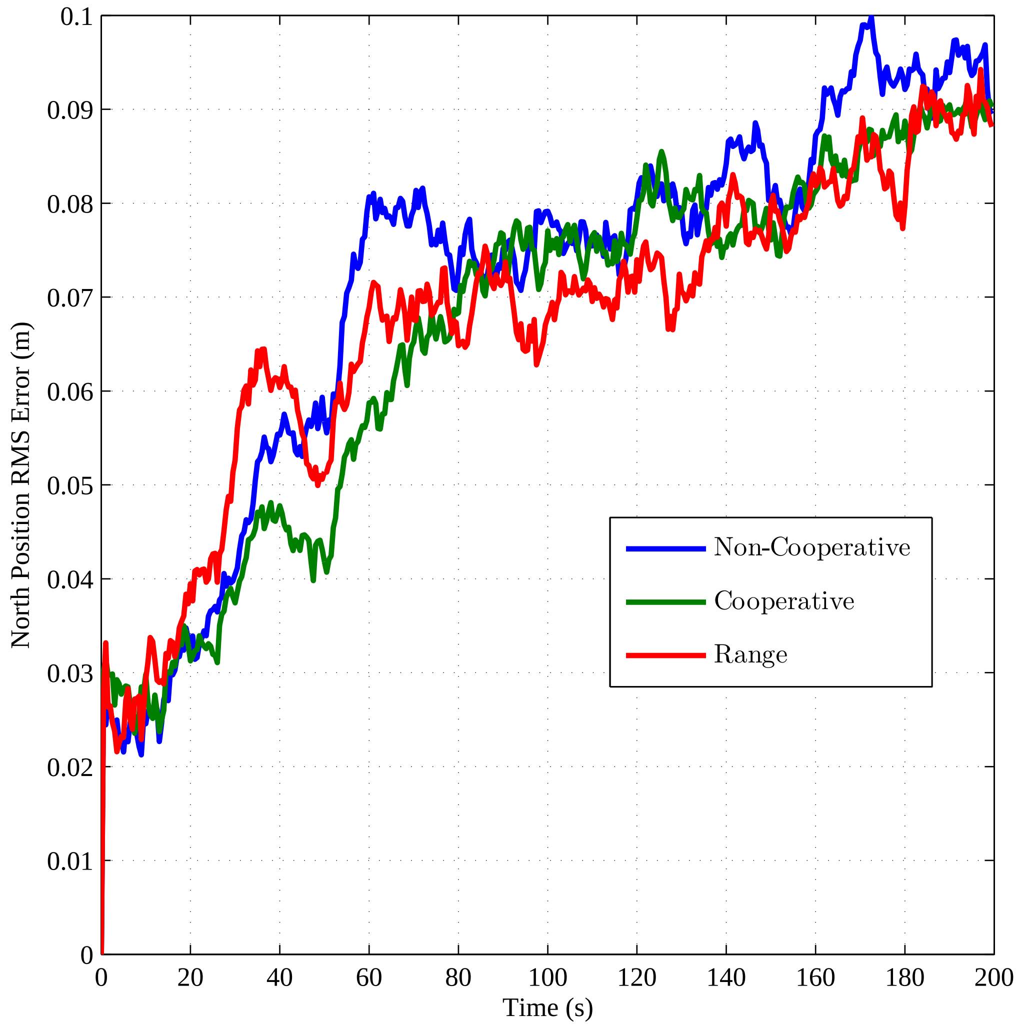


Figure E.1: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

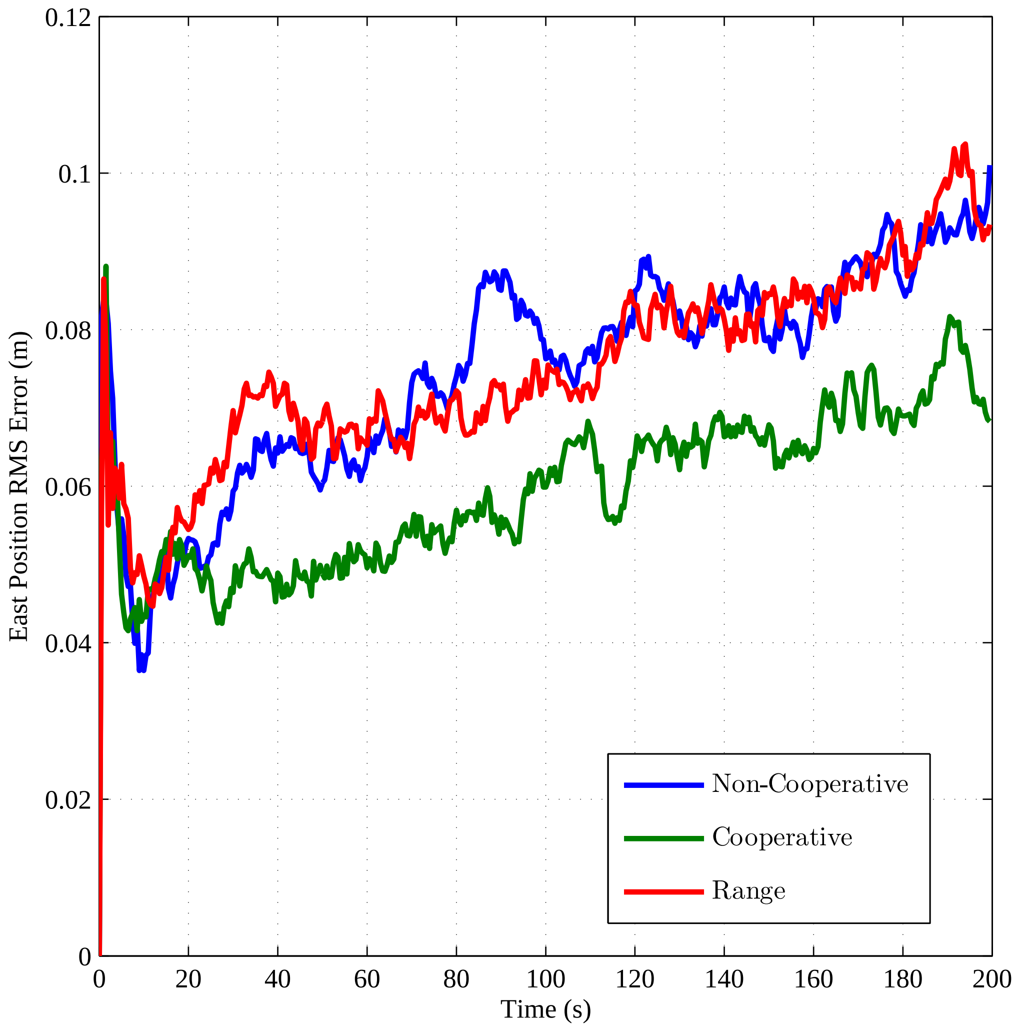


Figure E.2: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the east axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a 10 m radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

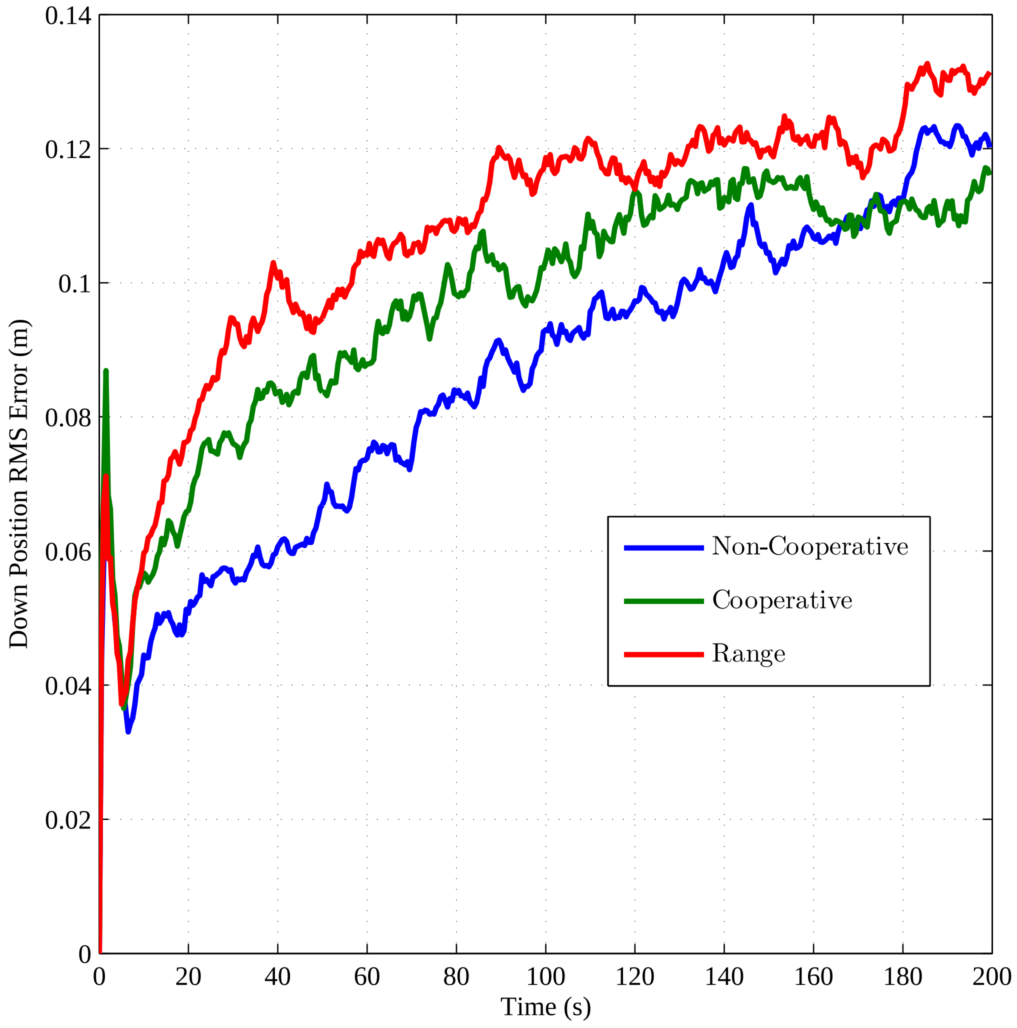


Figure E.3: Simulated 50-run Monte Carlo root-mean-squared (RMS) position error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a 10 m radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

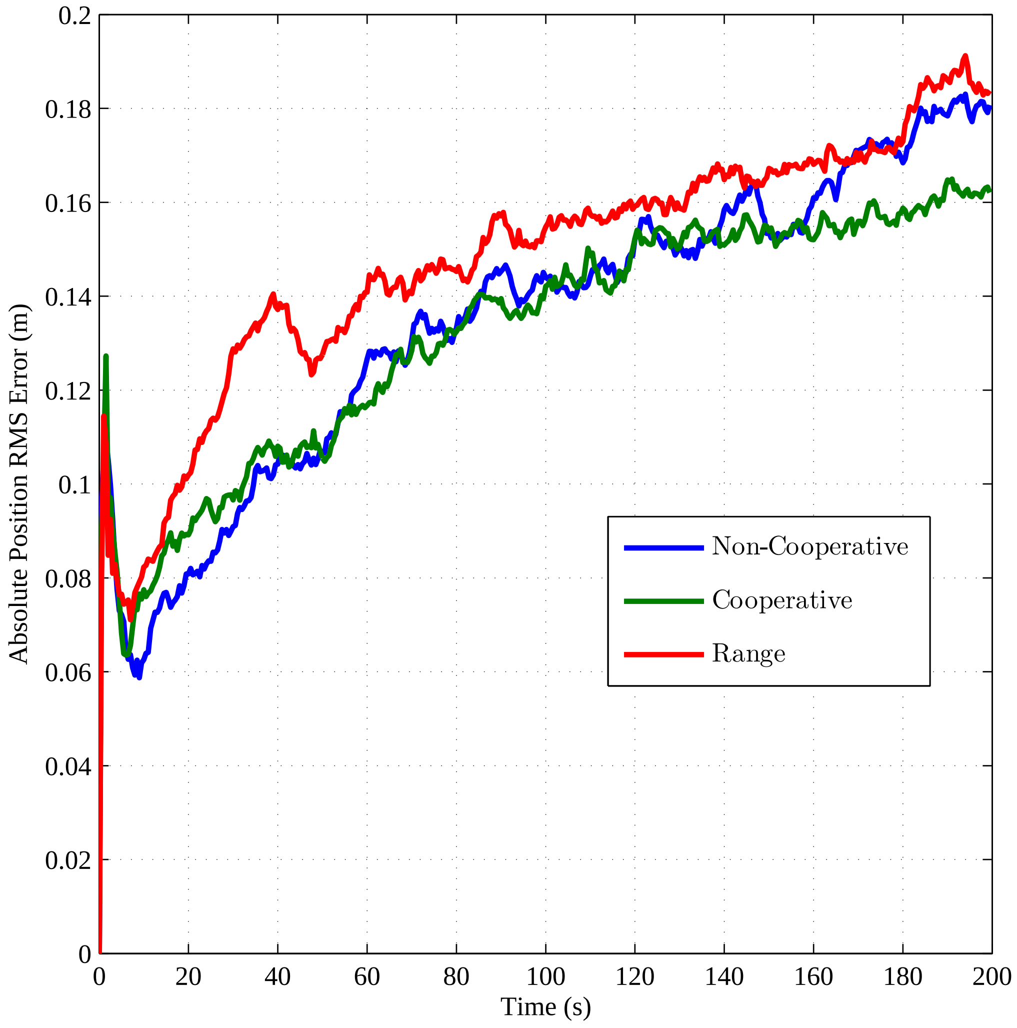


Figure E.4: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute position error results. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

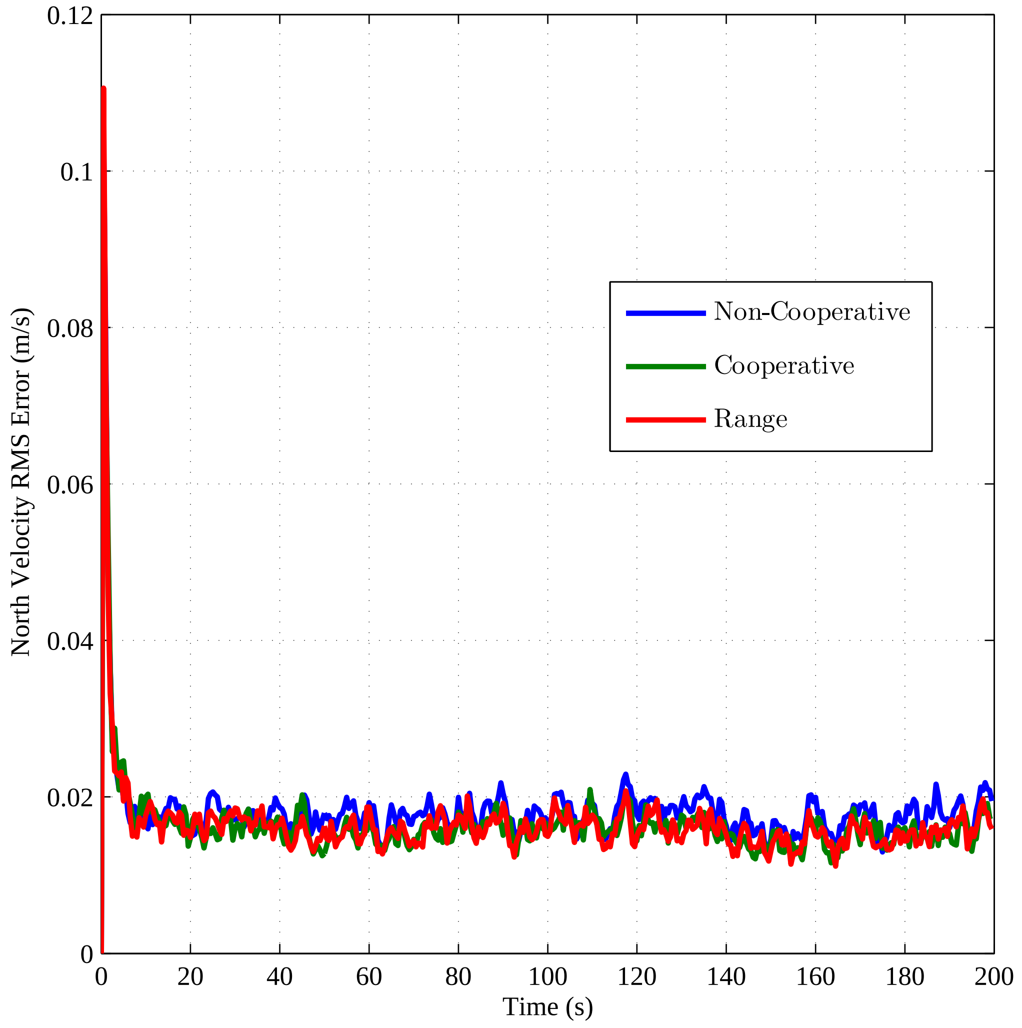


Figure E.5: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the north axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a 10 m radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

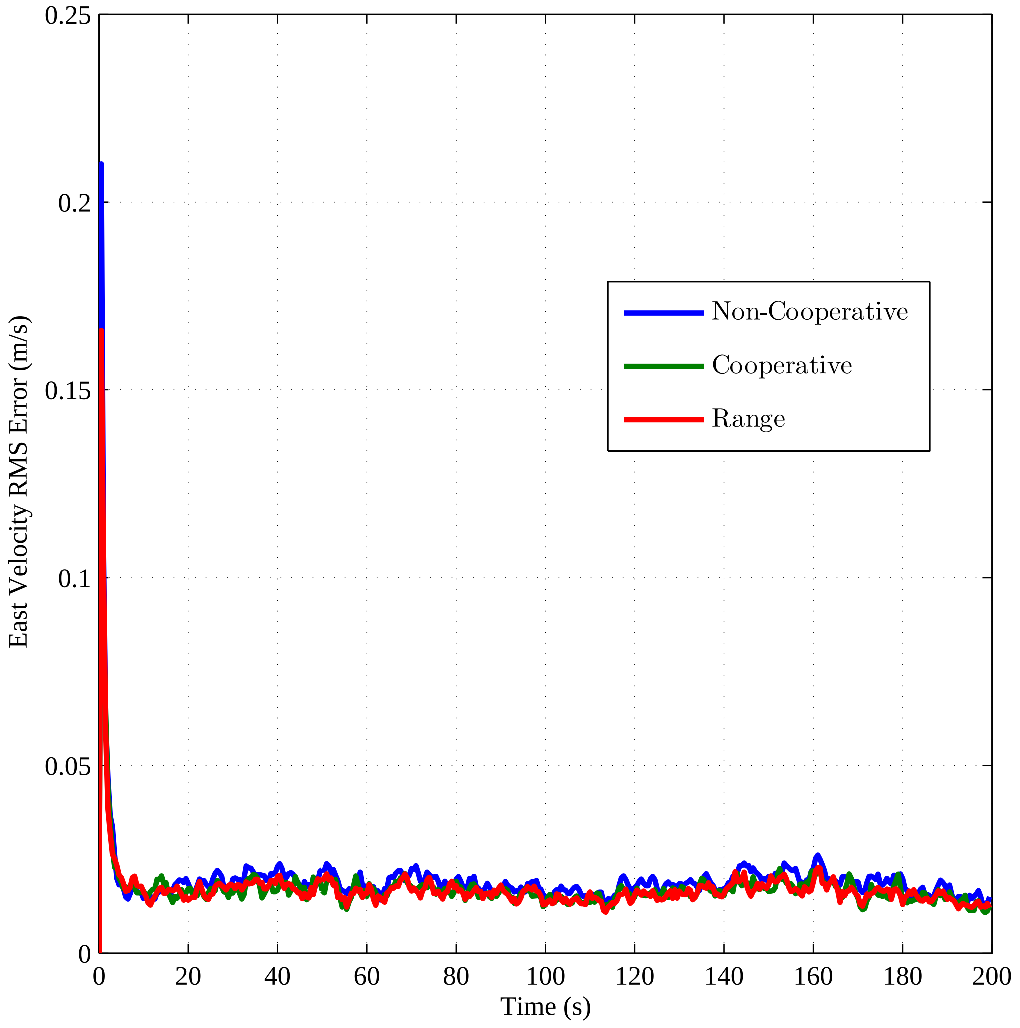


Figure E.6: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the east axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a 10 m radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

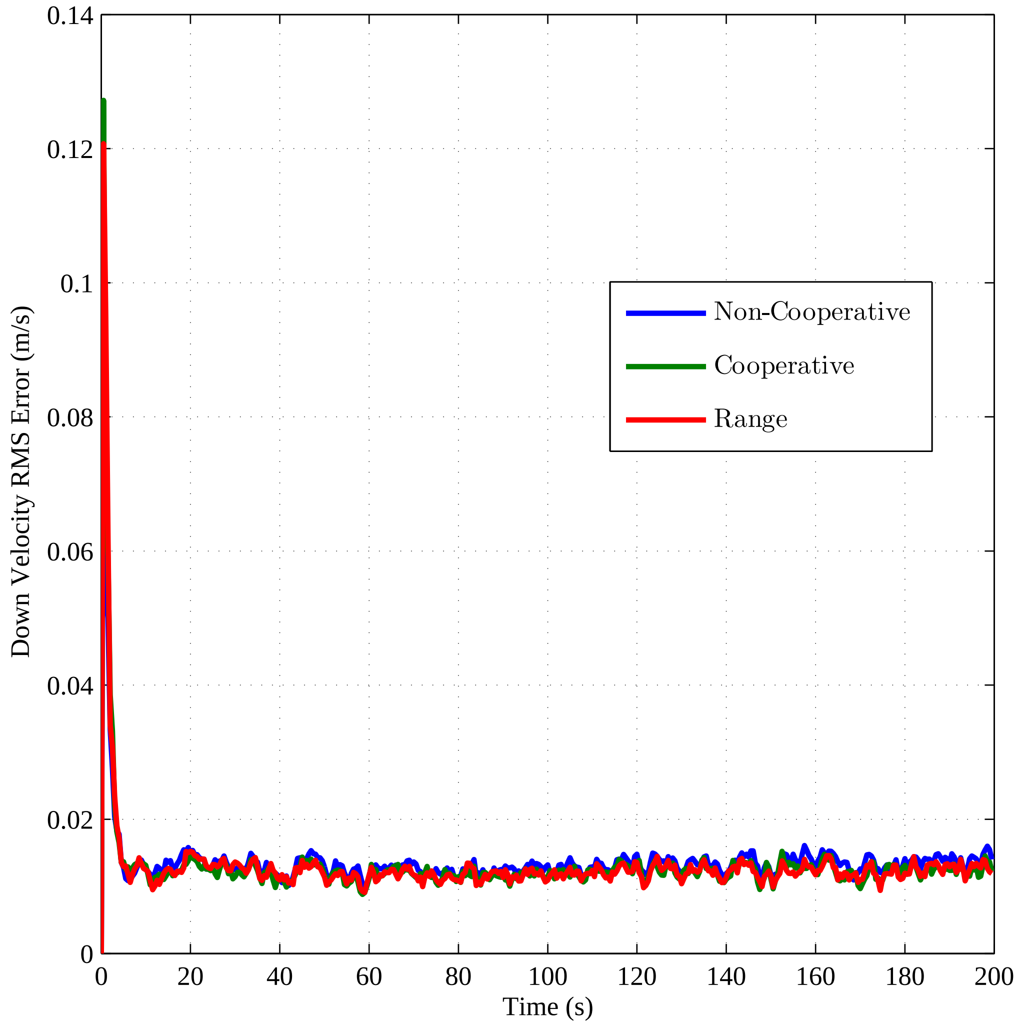


Figure E.7: Simulated 50-run Monte Carlo root-mean-squared (RMS) velocity error results along the down axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a 10 m radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

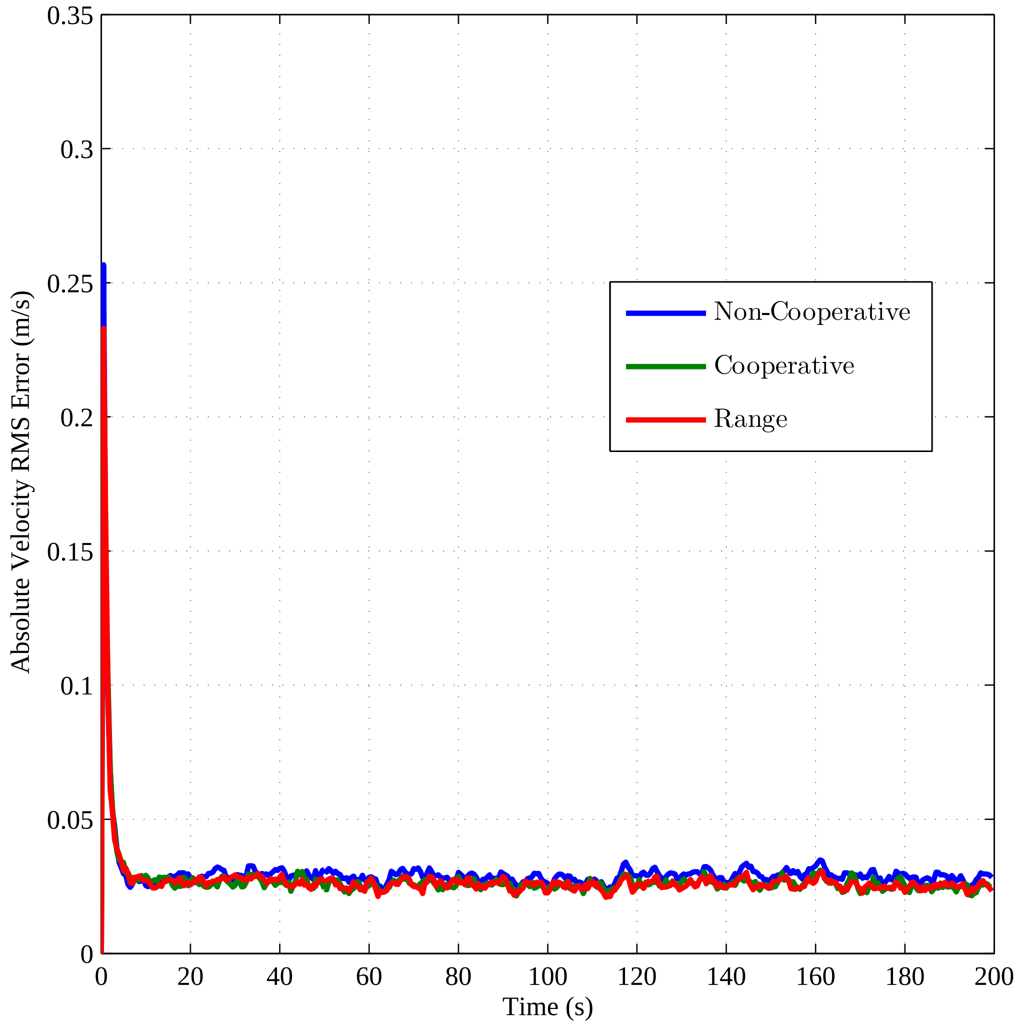


Figure E.8: Simulated 50-run Monte Carlo root-mean-squared (RMS) absolute velocity error results . The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

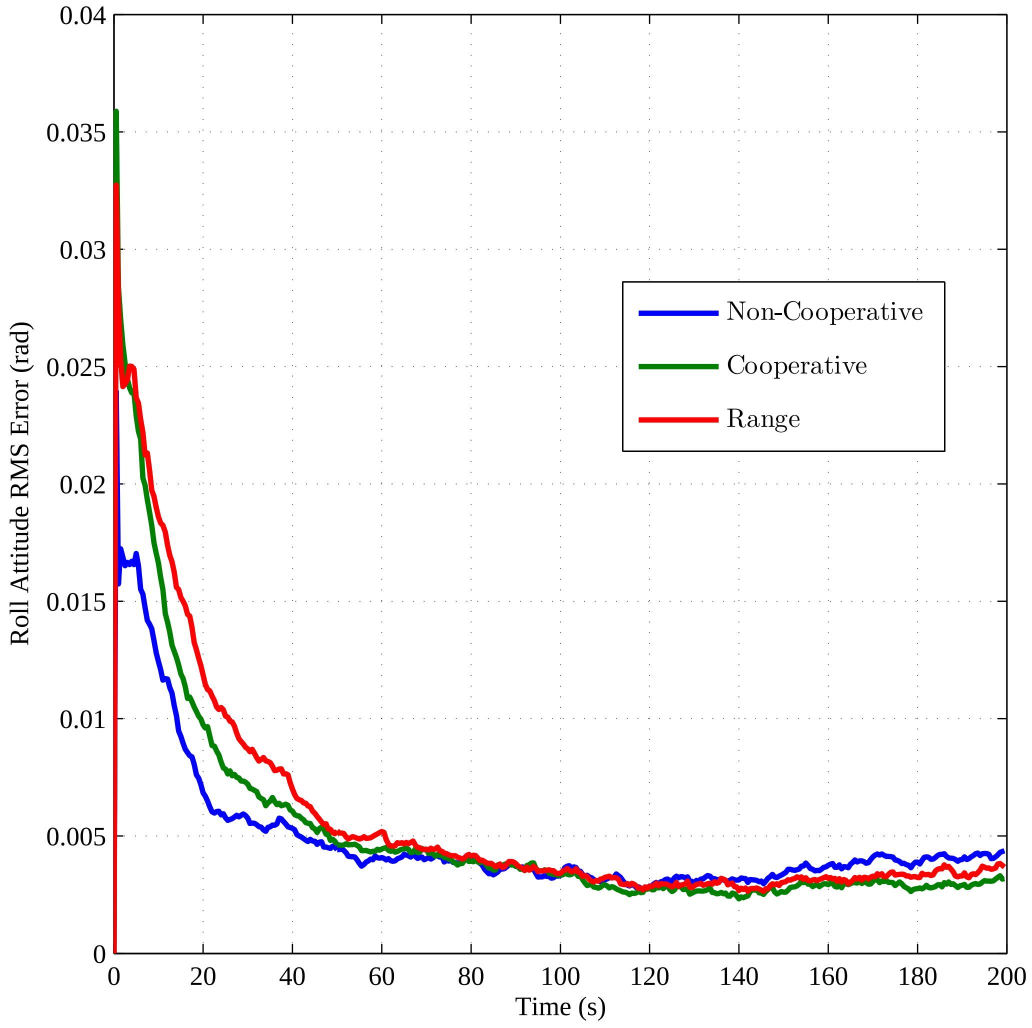


Figure E.9: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the north axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a 10 m radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

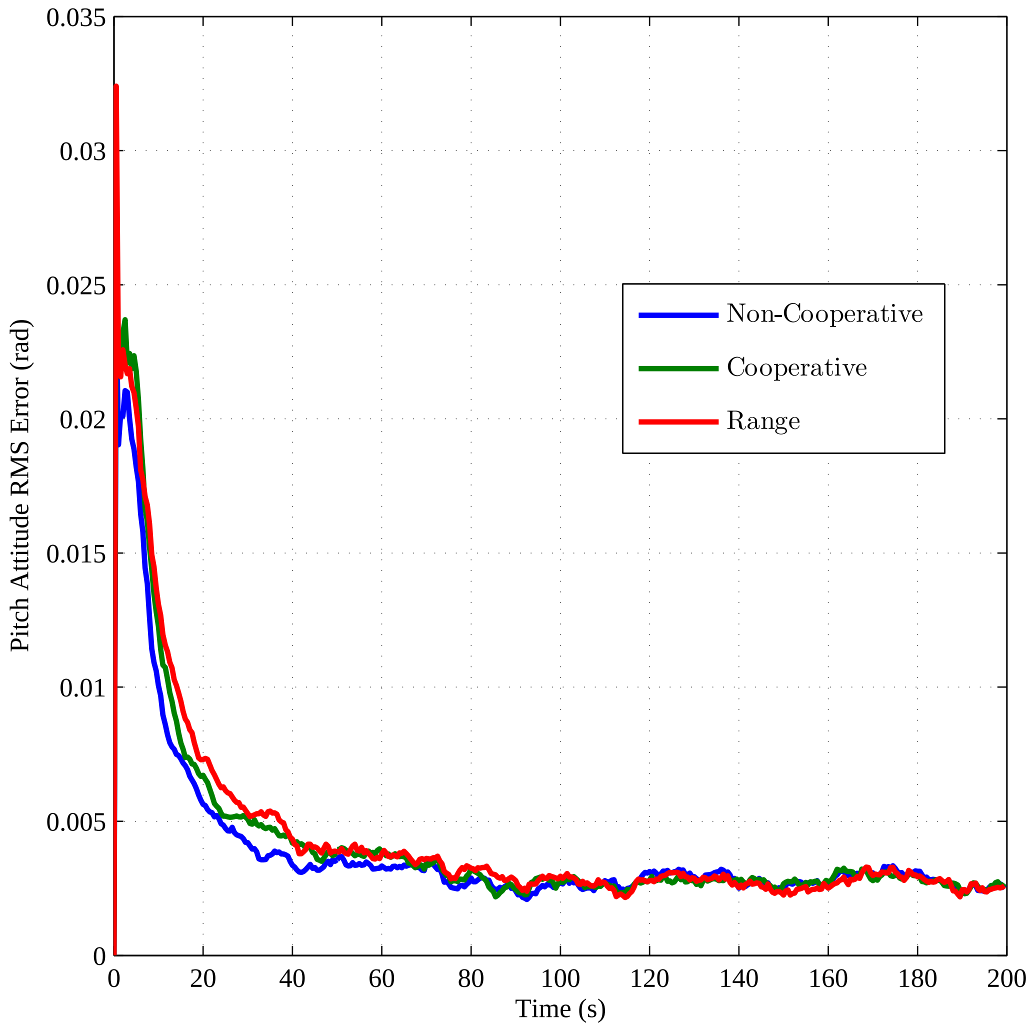


Figure E.10: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the east axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

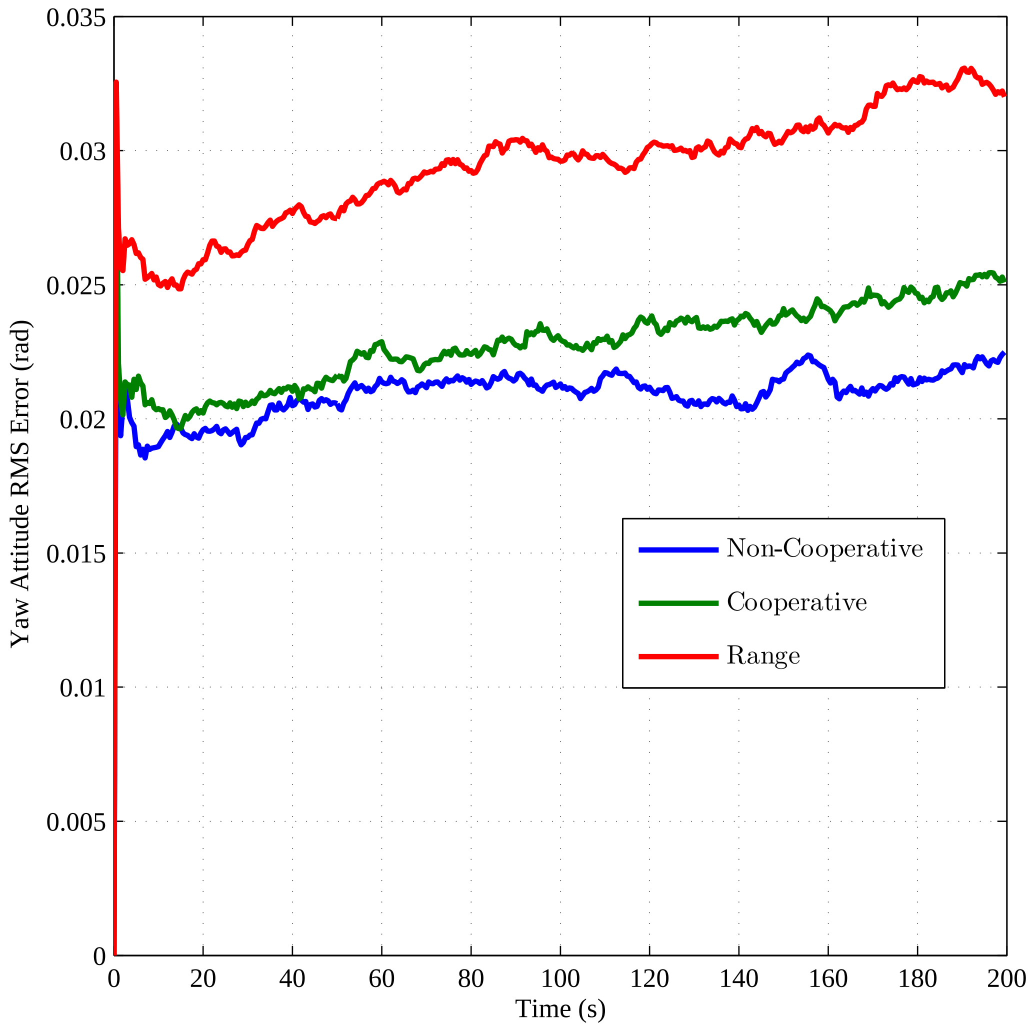


Figure E.11: Simulated 50-run Monte Carlo root-mean-squared (RMS) attitude error results about the down axis of the local navigation frame. The results shown are for the first of two platforms, performing a 360° counterclockwise yaw withing a $10\,m$ radius circular room, simulated using three system implementations of the pre-scaling method: 1) non-cooperative (blue), 2) cooperative without range measurements (green), and 3) cooperative with range measurements (red).

Bibliography

1. Baccou, P., B. Jouvencel, V. Creuze, and C. Rabaud. "Cooperative positioning and navigation for multiple AUV operations". *Oceans Conference Record (IEEE)*, 3:1816 – 1821, 2001. ISSN 0197-7385. URL `http://dx.doi.org/10.1109/OCEANS.2001.968122`.

2. Carlson, Neal A. *Distributed Kalman Filter Architectures, Phase II, Part A - Results*. Technical Report WL-TR-95-1096, Wright Laboratory, July 1997.

3. Carlson, Neal A. *Distributed Kalman Filter Architectures, Phase II, Part B - Appendices*. Technical Report WL-TR-95-1097, Wright Laboratory, July 1997.

4. Cook, Kendra L. B. "The Silent Force Multiplier: The History and Role of UAVs in Warfare". *Aerospace Conference, 2007 IEEE*, 2007:1 – 7, 2007.

5. Einstein, Albert and Robert W. Lawson. *Relativity: The special and general theory*. Routledge, 2001.

6. Gates, Robert M. "Remarks to Air War College (Maxwell, AL)". *Air War College*. April 21, 2008.

7. Geodesy, NIMA and Geophysics Department. *Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems*. Technical report, National Imagery and Mapping Agency, 2000.

8. Liu, Shirong, Linbo Mao, and Jinshou Yu. "Path Planning Based On Ant Colony Algorithm and Distributed Local Navigation for Multi-Robot Systems". *2006 IEEE International Conference on Mechatronics and Automation, ICMA 2006*, 2006:1733 – 1738, 2006. URL `http://dx.doi.org/10.1109/ICMA.2006.257476`.

9. Lowe, David G. "Distinctive Image Features from Scale-Invariant Keypoints". *International Journal of Computer Vision*, 60(2):91–110, 2004.

10. Madison, R., G. Andrews, P. Debitetto, S. Rasmussen, and M. Bottkol. "Vision-aided navigation for small UAVs in GPS-challenged environments". *Collection of Technical Papers - 2007 AIAA InfoTech at Aerospace Conference*, 3:2733 – 2745, 2007.

11. Mansour, Mohammad and Amir Shirkhodaie. "Navigational Task-Planning Of Distributed Cooperative Robotic Vehicles". *Conference Proceedings - IEEE SOUTHEASTCON*, 237 – 244, 2000. ISSN 0734-7502. URL `http://dx.doi.org/10.1109/SECON.2000.845570`.

12. Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, volume 1. Navtech, 1994.

13. Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, volume 2. Navtech, 1994.

14. Office of the Secretary of Defence. "Unmanned Systems Roadmap (2007-2032)". U.S. Department of Defense, December 2007.

15. Panzieri, Stefano, Federica Pascucci, and Roberto Setola. "Multirobot Localisation Using Interlaced Extended Kalman Filter". *IEEE International Conference on Intelligent Robots and Systems*, 2816 – 2821, 2006. URL `http://dx.doi.org/10.1109/IROS.2006.282065`.

16. Pedduri, Satish, K. Madhava Krishna, and Henry Hexmoor. "Cooperative navigation function based navigation of multiple mobile robots". *2007 International Conference on Integration of Knowledge Intensive Multi-Agent Systems, KIMAS 2007*, 277 – 282, 2007. URL `http://dx.doi.org/10.1109/KIMAS.2007.369822`.

17. Raghavan, Sriram and B. Ravindran. "Profiling pseudonet architecture for coordinating mobile robots". *Proceedings of the 2007 2nd International Conference on Communication System Software and Middleware and Workshops, COMSWARE 2007*, 4267994 –, 2007. URL `http://dx.doi.org/10.1109/COMSWA.2007.382570`.

18. Sanderson, Arthur C. "Distributed Algorithm for Cooperative Navigation among Multiple Mobile Robots". *Advanced Robotics*, 12(4):335 – 349, 1998. ISSN 0169-1864.

19. Titterton, David H. and John L. Weston. *Strapdown Inertial Navigation Technology*. The Institution of Electrical Engineers and The American Institute of Aeronautics and Astronautics, second edition, 2004.

20. Veth, Michael J. *Fusion of Imaging and Inertial Sensors for Navigation*. Ph.D. thesis, Air Force Institute of Technology, 2006.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)* 19-03-2009	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* September 2007-March 2009

4. TITLE AND SUBTITLE Vision-Aided, Cooperative Navigation for Multiple Unmanned Vehicles	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Bingham, Jason K., Captain, USAF	**5d. PROJECT NUMBER** 09-224
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765 DSN: 785-3636	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/09-05
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory (AFMC) Attn: Dr. T.J. Klausutis 101 W. Eglin Blvd., Suite 212 Eglin AFB, FL 32542-6810 timothy.klausutis@eglin.af.mil DSN: 875-0887 Comm: (850) 883-0887	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RW
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The motivation of this research is to exploit three attributes of increased unmanned vehicle use for intelligence, surveillance, and reconnaissance missions. These attributes are: increased numbers of unmanned vehicles, on-board vision, and wireless communications. The research begins with the development of a cooperative navigation system based on the measurement of vehicle position relative to shared landmark position estimates. Each vehicle in the network locates landmarks using it's on-board vision system and transmits the data to all other system vehicles. After receiving data from the other vehicles, the system fuses the landmarks with on-board measurements using a federated filter architecture. Simulations of the cooperative system, with and without ranging, are compared to a non-cooperative simulation. The comparison is performed using four platform motion scenarios: stationary, linear, angular, and full motion. The simulation results demonstrate position error estimate improvements of 0.5*cm* to 1*cm*. Additionally; the stationary and linear motion scenarios demonstrate attitude observability difficulties eliminated by the introduction of angular motion.

15. SUBJECT TERMS

navigation, image-aided navigation, cooperative navigation, federated filter, feature tracking, inertial navigation, passive navigation, autonomous navigation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** LtCol Michael Veth (ENG)
REPORT U	**ABSTRACT** U	**c. THIS PAGE** U	UU	210	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636 x4541 email: michael.veth@afit.edu

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18